VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2013 **Bc. Janusz Supik**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozšíření vhodné implementace směrovacího protokolu OSPF o možnost transportu TLV položek

Extension of OSPF Routing Protocol to Transport Type-Length-Value Records

2013 **Bc. Janusz Supik**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Janusz Supik**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozšíření vhodné implementace směrovacího protokolu OSPF o možnost transportu TLV položek
Extension of OSPF Routing Protocol to Transport Type-Length-Value Records

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat rozšíření směrovacího protokolu OSPF o možnost distribuce obecných TLV položek mezi softwarovými přepínači realizovanými na platformě GNU/Linux.

1. Seznamte se s problematikou distribuce směrovací informace a možnostmi šíření obecných TLV záznamů v rámci počítačové sítě provozující směrovací protokol OSPF.
2. Navrhněte a implementujte rozšíření tohoto protokolu o možnost propagace obecných TLV položek a to včetně logiky jejich zpracování na jednotlivých směrovačích.
3. Promyslete a na jednoduchém příkladě demonstrujte možnosti provázání navrženého řešení s externími zdroji informací.
5. Ověřte funkčnost řešení na konkrétní síťové topologii.

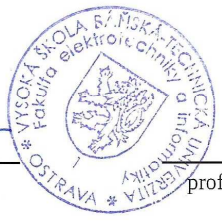
Seznam doporučené odborné literatury:

SCHRODER, Carla. Linux Networking Cookbook. USA : O Reilly Media, Inc., 2007. 640 s. ISBN 978-0-596-10248-7.
MOY, John T. OSPF: anatomy of an Internet routing protocol. USA : Addison-Wesley, 1998. 368 s. ISBN 978-0-201-63472-3.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Milata**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 11.4.2013 . . . . . .

podpis

Poděkování

Touto cestou bych rád poděkoval především **Ing. Martinu Milatovi** za odborné vedení, neocenitelné rady a připomínky k vypracování této diplomové práce. Také bych rád poděkoval celé rodině a mé přítelkyni za morální podporu při tvorbě této práce.

Abstrakt

Tato diplomová práce se zaměřuje na analýzu využití směrovacího protokolu OSPF jako prostředku pro distribuci uživatelských dat. Práce využívá jako základ volně šiřitelnou implementaci protokolu OSPF - Quagga. Výsledný program poskytuje funkčnost pro spojení a synchronizaci s externím zdrojem dat s minimální zátěží síťového provozu. První část práce je zaměřená na analýzu funkčnosti protokolu OSPF, jelikož znalost protokolu je nezbytná pro vývoj aplikace. Další důležitou částí práce je analýza možností implementace a samotný popis vybrané metody řešení. Poslední částí práce je testování mé výsledné implementace na konkrétní síťové topologii.

Klíčová slova: TCP/IP, sítě, směrovač, směrování, OSPF, Link State Advertisement, TLV položka, Quagga

Abstract

This diploma thesis is focused on analysis of using routing protocol OSPF as medium for transport of user data. Project is based on open source implementation of OSPF protocol - Quagga. Final program provides functionality for connect with external data source with minimal load of network. First part of work is focused on functionality analysis of the OSPF protocol because knowledge of the protocol is necessary for application development. Next important part is analysis of implementation options and description of chosen implementation method. Last part is testing of my application on real network topology.

Keywords: TCP/IP, networks, router, routing, OSPF, Link State Advertisement, TLV data, Quagga

Seznam zkratek

ASBR	Autonomous System Boundary Router
AS	Autonomous System
BDR	Backup Designated Router
BGP	Border Gateway Protocol
BSD	Berkeley Software Distribution
DDP	Database Description Packet
DR	Designated Router
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
LS	Link State
LSA	Link State Advertisement
LSAck	Link State Acknowledge
LSDB	Link State Database
LSID	Link State Identification
LSR	Link State Request
LSU	Link State Update
MAC	Media Access Control
MPLS	MultiProtocol Label Switching
MTU	Maximum Transmission Unit
NBMA	Non Broadcast Multiple Access
OS	Operating System
OSPF	Open Shortest Path First
PDU	Protocol Data Unit
RIP	Routing Information Protocol
SPF	Shortest Path First
TCP	Transmission Control Protocol
TLV	Type Length Value
ToS	Type of Service
TTL	Time To Live

Obsah

1. Úvod

Sítě založené na protokolu TCP/IP se využívají po celém světě. Jsou nasazovány jak ve velkých komerčních sítích, tak i v malých domácích prostředích. Při spojování různých síťových segmentů je nutno mezi nimi datový tok směrovat. Směrování probíhá na základě dat uložených ve směrovací tabulce. Tyto data můžeme v tabulce definovat pomocí konfigurace staticky nebo je získávat z dynamických směrovacích protokolů.

Existují dva hlavní způsoby možnosti nasazení směrovacích protokolů. První z nich je využití specializovaného hardwaru - směrovače, který tuto funkčnost poskytuje. Druhou variantou je využití běžně dostupného zařízení, jako je např. PC, na kterém je nutno provozovat směrovací software. Mezi tyto směrovací protokoly se řadí i protokol OSPF.

Existuje několik volně šiřitelných implementací protokolu OSPF. Nicméně žádná z nich neposkytuje funkčnost pro přenos libovolných uživatelských dat. Obsahují pouze základní mechanismy distribuce uživatelských dat, ale žádnou konkrétní logiku zpracování uživatelem definovaných dat.

Mezi nejpoužívanější volné implementace protokolu OSPF se řadí Quagga, Zebra a Vyatta VC. Aplikaci Zebra jsem vyřadil ze svého výběru jelikož je to již několik let nevyvíjený projekt. Z projektu Zebra vznikl projekt Quagga, která má stále aktivní komunitu vývojářů a uživatelů. Posledním projektem je Vyatta VC, který je operačním systémem, nejen samostatnou aplikací. Ovšem jeho část řídící směrování je právě projekt Quagga, tudíž jsem se rozhodl zvolit jako výchozí aplikaci právě Quaggu.

V první části se budu věnovat problematice protokolu OSPF, tuto část rozdělím na dvě samostatné kapitoly. V první kapitole této části se budu věnovat stručnému popisu protokolu OSPF jako jednotného funkčního celku. Ve druhé kapitole se budu podrobně věnovat jednotlivým částem protokolu, které jsou nutné pro jeho korektní činnost. Druhou část textu, zabývající se samotnou implementací požadované funkčnosti, rozdělím do tří kapitol. V první kapitole této části se budu věnovat stručné charakteristice vybrané implementace protokolu OSPF - Quagga. Popíši její základní vlastnosti, způsob implementace a základní konfiguraci instalace. V další kapitole popíši samotnou implementaci a nutné úpravy kódu Quaggy, aby poskytovala požadovanou funkčnost. V poslední kapitole se chci věnovat otestování mnou navrženého a implementovaného řešení. Provedu analýzu provozu mezi směrovacími procesy Quaggy.

2. Úvod do protokolu OSPF

Technickým a provozním detailům protokolu OSPF se budu věnovat v následující kapitole. V této kapitole představím základní charakteristiky a vlastnosti protokolu, které stručně popíši.

OSPF je klasifikován jako IGP a řadí se do třídy protokolů stavu linky (link-state). U tohoto typu směrovacích protokolů si každý směrovač uchovává databází popisující kompletní topologii autonomního systému, případně oblasti, ve kterém se daný směrovač nachází. Tato databáze se nazývá databáze stavu linek (link-state database - LSDB, která obsahuje informace o směrovačích a propojovacích linkách mezi nimi). Databáze mají na každém směrovači lokální význam (aktivní rozraní konkrétního směrovače, aktivní sousedící směrovače). Databáze musí obsahovat kompletní informace o topologii z důvodu rychlé konvergence při změně v síti (např. výpadek linky nebo celého směrovače). Negativním efektem nutnosti uložení databáze celé sítě je bezesporu paměťová náročnost, navíc potřeba výpočtů cest topologie zvyšuje zatížení procesoru směrovače.

Pomocí zaplavení sítě informacemi o připojených linkách (flooding) šíří směrovač tuto svou databázi do celého autonomního systému, ve kterém se nachází. Z těchto databází si každý směrovač vypočítá stromovou topologii sítě, ve které je kořenem. Tento strom je počítán na základě SPF algoritmu - cesty k jednotlivým uzlům jsou nejkratší možné, výpočet je prováděn Dijkstrovým algoritmem. Tento strom udává cestu ke všem uzlům daného autonomního systému. Jednotnost výpočtů těchto stromů je zaručená tím, že všechny směrovače využívají stejný algoritmus výpočtu nad stejnými daty. Externě odvozené směrovací informace (například z EGP, jako je BGP) jsou v stromu topologie uložené jako listy. Navíc tyto data jsou oddělená od ostatních dat stavu linek, která jsou využívána protokolem OSPF.

Výměny dat databází u protokolu OSPF mohou být ověřovány, pouze důvěřované (směrovače kódující směrovací informace stejným klíčem) směrovače se mohou podílet na směrování v daném autonomním systému. Mohou být využívány různé ověřovací metody v každé IP podsíti.

Protokol OSPF poskytuje flexibilní nástroj konfigurace podsítí - každá cesta distribuovaná protokolem má IP adresu a masku sítě. Každá podsíť může mít různou velikost (v závislosti na velikosti masky sítě). Paket je směrován pomocí nejlepšího (nejdelší shoda adresy sítě) záznamu ve směrovací tabulce (o to se již stará směrovač, nikoli protokol OSPF).

Protokol poskytuje možnost spojení několika sítí do jednoho celku zvaného oblast (area). Tyto oblasti mají směrování oddělené od zbytku autonomního systému, což omezuje přenos směrovacích informací sítí - šetří kapacitu linek. [1]

2.1. Důležité pojmy

V textu jsem již využil některé pojmy, které jsou často využívány v kontextu protokolu OSPF v této práci. V této části bych je rád přiblížil.

- **Směrovač -** aktivní prvek sítě, pracující na třetí vrstvě IP. Jeho funkcí je směrovat pakety do cílových sítí (směrovat je podle cílové IP adresy).
- **ID směrovače -** 32 bitové číslo, které je přidělené každému směrovači, na kterém je provozován protokol OSPF. Toto číslo je unikátní v rámci jednoho autonomního systému. Jeden směrovač může mít přidělená různá ID, která jsou vázána na různé směrovací protokoly (OSPF, BGP, EIGRP).
- **IP síť -** jedna fyzická síťová topologie může obsahovat více adres sítě/podsítě. Dělení topologie na IP sítě je zajištěno pomocí směrovačů.
- **Síťová maska -** 32 bitové číslo uvádějící, která část IP adresy je adresou sítě, a která adresou zařízení. Na začátku masky se nachází sekvence jedniček (ty udávají adresu sítě), která je následována sekvencí nul (ty udávají adresu zařízení, které doplní masku do délky 32 bitů.
- **Autonomní systém -** z hlediska OSPF skupina směrovačů, které mezi sebou sdílejí směrovací informace obvykle pomocí interního směrovacího protokolu. Bývá označován jako AS.
- **Interní směrovací protokol (IGP) -** takový směrovací protokol, který funguje na směrovačích uvnitř autonomního systému. Každý autonomní systém provozuje pouze jeden typ směrovacího protokolu. Různé autonomní systémy mohou pracovat s rozdílným interním směrovacím protokolem.
- **Rozhraní -** spojení směrovače s některou z jeho připojených sítí. Stavové informace rozhraní jsou získávány z komunikačních protokolů nižších síťových vrstev a ze směrovacího protokolu. Rozhraní k síti má svou přidělenou IP adresu a masku sítě.
- **Sousedící směrovače a jejich vztahy (sousedství a přilehlost) -** sousedící směrovače mají svá rozhraní připojená do společné části sítě. Jejich vztah sousedství se navazuje a udržuje pomocí části OSPF - Hello protokolu. Vztah sousedství je navázán mezi všemi směrovači v síťovém segmentu. Vztah přilehlosti se naváže až po synchronizaci databází směrovačů. Vztah přilehlosti se nemusí navazovat mezi všemi směrovači, které mají mezi sebou sousedskou vazbu. V některých typech sítí (broadcast, NBMA) se vztah přilehlosti navazuje pouze mezi některými směrovači. Detailnější popis uvedu dále.
- **Hello protokol -** část protokolu pro nalezení sousedů, navazování a udržování sousedských vztahů.
- **Zaplavení -** mechanismus obsažený v OSPF protokolu, který se stará o distribuci databází stavu linek mezi OSPF směrovači. [9]

3. Detaily fungování protokolu OSPF

Protokol OSPF funguje jako celek skládající se z několika částí, které spolu spolupracují. Mezi tyto hlavní části patří: Hello protokol, LSA a SPF proces pro výpočet nejkratších cest. Funkce jednotlivých částí budou uvedeny dále.

Pomocí Hello protokolu se navazují sousedské vztahy mezi spojenými směrovači. V některých případech si sousedské směrovače vymění své databáze a jejich vztah se stává bližší, tyto směrovače se nazývají přilehlé (adjacency).

LSA mechanismus slouží pro získávaní, udržování a aktualizaci jednotlivých databází stavu linek. Pro svou činnost potřebuje Hello protokol pro objevení sousedských směrovačů. Poslední částí, která úzce spolupracuje s LSA je proces pro výpočet nejkratších cest (SPF). [7]

3.1. Metrika

Jako hlavní parametr u výpočtů nejkratších (nejvýhodnějších) cest do cílové sítě se používá kritérium, označované jako metrika. Protokol OSPF využívá metriku, která se označuje jako cena (cost). Udává se jako celé číslo z rozsahu 1 až 65 535. Toto číslo se následně přiřadí ke každému rozhraní směrovače. Čím je toto číslo menší, tím má větší prioritu a cesta skrz toto rozhraní je více preferovaná.

Automaticky se cena vypočítává pomocí vzorce: cena = 10^8 / šířka pásma (v bps - bity za sekundu), cena je tedy závislá na rychlosti linky. Pro správnou funkci mechanismu je nutné, aby byly správně nastavené šířky pásma na jednotlivých rozhraních. Pro linky typu FastEthernet a rychlejší bude cena vždy rovná 1. Existuje možnost konstantu 10^8 při výpočtu ceny linky nahradit za vyšší, tuto možnost nepodporují všechny implementace protokolu.

Ovšem cenu je možno nastavit i staticky. V případě využívání pouze linek s nejvyšší prioritou, by mohlo docházet k jejich přetěžování. Je tedy vhodné někdy nastavit rychlejším linkám vyšší cenu a tím zabránit přetěžování daných linek. V některých cestách se pak použije pomalejší linka a tak se uvolní pásmo na rychlejší. [2]

3.2. Typy sítí z hlediska OSPF

Z hlediska fungování protokolu OSPF se sítě, v závislosti na jejich realizaci, dělí na několik typů. Rozdíly se projeví nejvíce v těchto ohledech funkčnosti:

- **Navazování a udržování vztahů sousednosti** - OSPF vždy využívá Hello protokol, ale ten funguje na každém typu sítě odlišně.
- **Synchronizování databází stavu linek -** liší se ve způsobu distribuce směrovacích informací.
- **Abstrakce** - liší se v reprezentaci topologie sítě v databázi.

3.2.1. Broadcast sítě

V brodcast síti je každý komunikující uzel schopný odeslat paket, který bude přijat všemi ostatními uzly. Typický představitel je síť typu Ethernet. Zvláštním typem paketu je broadcast paket odesílaný na adresu 0xffffffffffff. Tento paket přijímají všechny stanice.

Na jednom segmentu sítě může být i více směrovačů. Z tohoto důvodu se volí DR a BDR, se kterými zbylé směrovače navazují vztahy přilehlosti. Vztah mezi směrovači, které nejsou DR ani BDR zůstane ve stavů Two-way. [3]

3.2.2. NBMA sítě

V síti tohoto typu je možno spojit více než dva směrovače. Síť ovšem neumožňuje posílání broadcastových paketů, které by přijaly všechny směrovače současně. V tomto typu sítí se volí DR a BDR a následné komunikace probíhá jednotlivě mezi směrovači pomocí unicastové komunikace.

Pokud síť nebude v zapojení full-mesh můžou mít směrovače problém s nalezením svých sousedů. Z tohoto důvodu se sousední směrovače definují při konfiguraci prvků. Komunikace směrovačů s DR nebo BDR probíhá prostřednictvím virtuálních okruhů. [2]

3.2.3. Point-to-point a point-to-multipoint sítě

Jako point-to-point sítě se označují sítě, které spojují pouze dva zařízení. Nejčastějším zástupcem jsou sériové linky. Nevolí se DR ani BDR. Směrovače komunikují pomocí multicastu.

Jako point-to-multipoint lze chápat zvláštní případ NBMA sítě. Síť je považovaná za sadu point-to-point linek, proto se nevolí DR ani BDR a komunikace probíhá pomocí multicastu. [2]

3.3. Enkapsulace OSPF paketů

Ještě před popisováním samotných principů funkčnosti a přenášení dat se zmíním o způsobu enkapsulace OSPF paketů do protokolu síťové vrstvy.

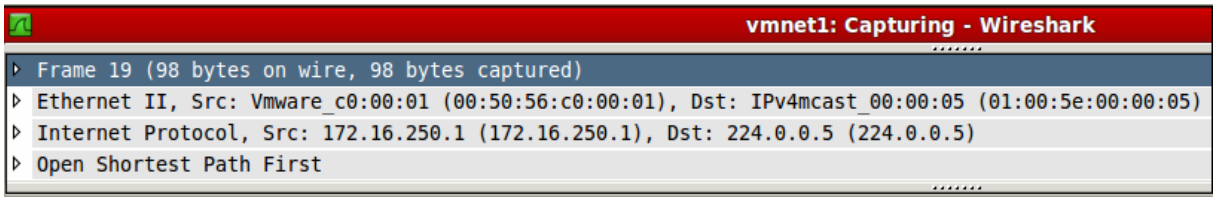
Pakety protokolu OSPF se transportují pomocí IP protokolu na síťové vrstvě. Nevyužívá se transportních protokolů TCP nebo UDP. Pakety jsou enkapsulovány přímo do IP protokolu s hlavičkami lokálních datových linek. Obrázek 1 ukazuje zachycenou zprávu OSPF.

OSPF nekontroluje velikost své datové jednotky. Nemá definovaný mechanismus fragmentace, proto musí využívat schopnost fragmentace IP, když odesílá nebo přijímá zprávy

delší, než je MTU sítě. Bez ztráty funkcionality se musí fragmentovat velké OSPF pakety (DDP, LSR, LSU, LSAck). Největší problém nastává u virtuálních linek kde je limit velikosti paketů nastaven na 576 bytů. V některých případech může maximální velikost IP paketu dosahovat až 65 536 bytů (včetně hlavičky IP).

OSPF může pro některé své zprávy využívat multicast, pokud jsou posílány na síť s vícenásobným přístupem (broadcast, NBMA). Protokol využívá dvě cílové multicastové adresy. Pakety zaslané na tyto adresy nejsou určeny k směrování, tudíž se jim hodnota TTL nastaví na 1. Využívané adresy:

- **AllSPFRouters** tato adresa má přidělenou hodnotu 224.0.0.5. Na tuto adresu se zasílají všechny hello zprávy a některé zprávy využívané při distribuci směrovacích informací (flooding). Na této adrese by měly být schopny komunikovat všechny směrovače.
- **AllDRouters** tato adresa má přidělenou hodnotu 224.0.0.6. Některé pakety při distribuci směrovacích informací využívá tuto adresu. Na této adrese musí naslouchat všechny DR a BDR.



Obrázek 1. Enkapsulace datové jednotky OSPF

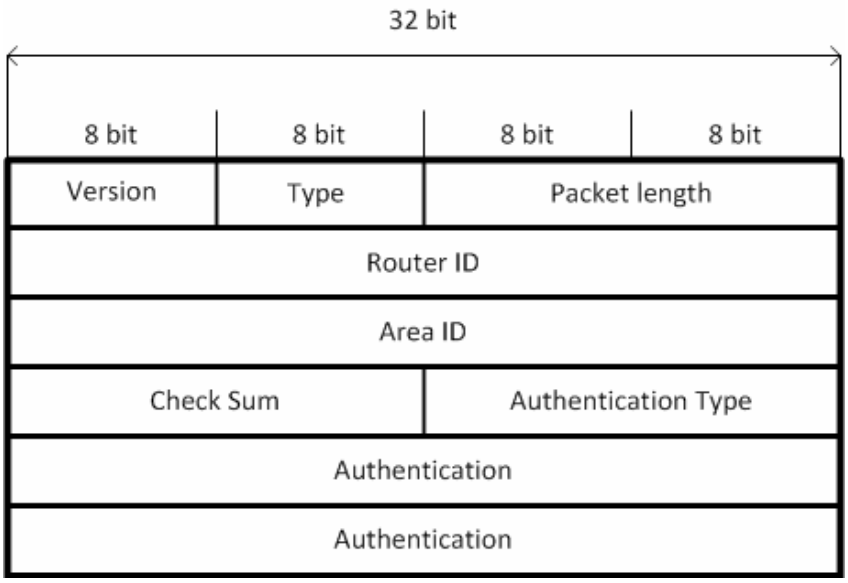
V záhlaví IP paketu se protokol OSPF označuje číslem 89. OSPF pakety se odesílají s IP ToS polem, přesněji IP precedence pole v ToS poli, nastaveným na hodnotu 110 - internetwork control. [1]

3.3.1. Hlavička OSPF paketu

Jako první údaj se v hlavičce nachází verze (pole Version) OSPF, v současně specifikaci se nejvíce vyskytuje OSPF verze 2. Hlavička OSPF paketu rovněž přenáší informaci o délce PDU. Pomocí pole **Type** se rozlišuje typ zprávy, kterou OSPF paket přenáší, rozlišujeme 5 hlavních typů zpráv:

- **Hello Packet** slouží pro vytvoření a udržování vztahu sousednosti mezi dvěma přilehlými směrovači. V broadcast sítích se využívá k volbě DR a BDR.
- **Database Description Packet (DDP)** popisuje aktuální databází stavu linek směrovače a podle tohoto stavu směrovač ověřuje zda má svou databází aktuální nebo si ji musí aktualizovat. Paket je vygenerován při vytváření vztahu přilehlosti.

- **Link State Request (LSR)** tímto paketem směrovač žádá aktualizaci své LSA databáze (v případě staré databáze nebo chybějících záznamů). Neaktuálnost záznamů může směrovač odhalit například pomocí DDP.
- **Link State Update (LSU)** je využíván k distribuci LSA sítí a také slouží jako odpověď na LSR paket.
- **Link State Acknowledgment (LSAck)** potvrzuje přijatá data pomocí LSU. Každé LSA musí být potvrzeno. V jednom paketu můžeme potvrdit více LSA, v LSR se posílají jen hlavičky jednotlivých LSA.



Obrázek 2. Hlavička OSPF zprávy

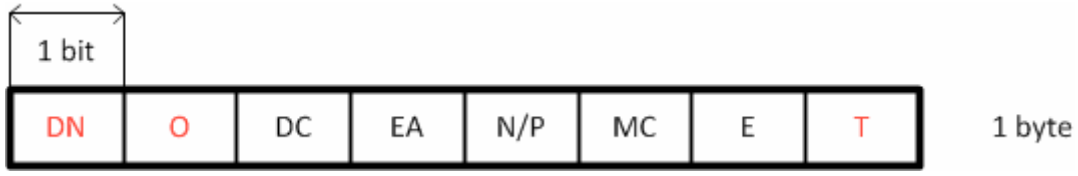
Obrázek 2. znázorňuje hlavičku paketu protokolu OSPF. Pole **RouterID** slouží pro jednoznačnou identifikaci směrovačů (volí se nejvyšší IP adresa z loopback rozhraní nebo nejvyšší IP adresa libovolného rozhraní), které je nezbytné pro fungování směrovacího protokolu. Pole **AreaID** se využívá pro identifikaci oblasti, ve které směrovač pracuje. V poli **CheckSum** je umístěn kontrolní součet paketu. Pole **AuthenticationType** určuje typ autentizace a posledních osm bytů hlavičky protokolu nese informace o autentizaci. [1]

Následuje popis části těla OSPF zpráv.

3.3.2. Options v OSPF zprávách

Toto pole se nenachází přímo v hlavičce OSPF zpráv, ale v jejich těle (payload). Pole je využíváno hello pakety a DDP. Hlavním důvodem jejich použití, je možnost poskytnout směrovačům využití dodatečných funkcí a použití těchto funkcí deklarovat ostatním směrovačům v síti. To umožňuje využití směrovačů s různou funkčností v rámci jedné směrované domény (AS nebo oblast).

Pět bitů pole má význam pevně definovaný ve standardu protokolu (dokumentu RFC2328). Zbylé tři jsou využívány pro stejnou funkci, ale ne všechny směrovače je mohou podporovat. Směrovače, které nepodporují jednotlivé funkčnosti (nerozeznají význam jednotlivých bitů), mají tyto bity nastavit na hodnotu 0. Na obrázku jsem tyto tři bity odlišil červenou barvou. Pořadí bitů je znázorněné na obrázku 3.



Obrázek 3. Pole options v OSPF

- **DC** bit určuje jak bude směrovač zacházet s okruhy vytvořenými externím požadavkem.
- **EA** bit označuje zda směrovač bude přijímat a přeposílat External-Attributes-LSA.
- **N/P** bit popisuje zacházení s LSA typem 7. Pokud je nastaven na 1, tak hodnota E bitu musí být nastavená na 0. Využívá se pouze u Hello zpráv.
- **MC** bit označuje povolení multicastového OSPF.
- **E** bit znamená povolení externích LSA v oblasti, ze které je příchozí paket.
- **DN** bit využívá se při MPLS/VPN v síti poskytovatele. Když je nastaven na hodnotu 1, tak LSA typu 3, 5 a 7 nejsou využity při výpočtu nejkratších cest.
- **O** bit označuje použití Opaque LSA. Využívá se pouze u DDP paketů. Pole je nezbytné pro funkčnost mých následujících úprav kódu.
- **T** bit indikuje použití směrování ToS.

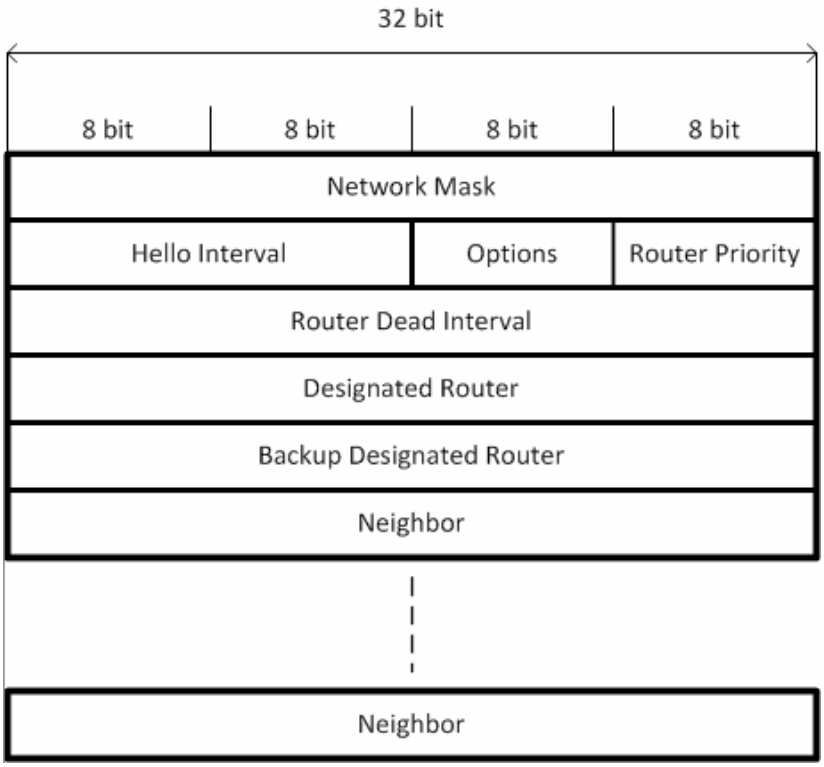
3.4. Hello protokol

Pro úspěšnou výměnu směrovacích informací mezi směrovači musí směrovač nejprve rozpoznat sousedící směrovače a navázat s nimi sousedský vztah. Až po navázání vztahu sousednosti si mohou směrovače posílat směrovací informace. Protokol poskytuje funkčnost pro objevení souseda, navázání vztahu, jeho udržování a možnost jeho ukončení.

Hello pakety se periodicky odesílají na všechny aktivní rozhraní, která využívají protokol OSPF. Způsobů odeslání je více. V broadcastových sítích a v sítích typu point-to-point se jako cílová adresa použije multicastová IP adresa AllSPFRouters. Všechny směrovače, které provozují protokol OSPF, musí přijímat pakety, které byly zaslány na tuto adresu. Na virtuálních linkách se hello pakety odesílají unicastově, přímo na adresu konce virtuální linky. V sítích point-to-multipoint se v daném intervalu zasílají hello pakety zvlášť na všechny směrovače, které jsou při konfiguraci umístěny do seznamu sousedních směrovačů. [7]

Obrázek 4. zobrazuje přenášenou datovou část hello zprávy. Význam jednotlivých polí:

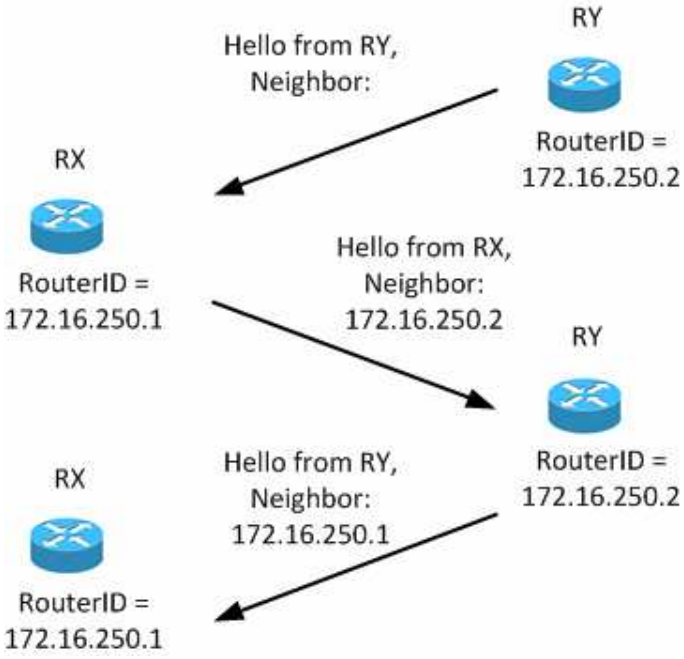
- **Network mask** reprezentuje masku sítě, ve které se nachází směrovač, který odeslal hello zprávu.
- **Hello interval** udává dobu mezi odesláním jednotlivých hello zpráv (v sekundách).
- **Options** udává zda směrovač využívá rozšiřující funkce protokolu OSPF.
- **Router priority** je využíváno v procesu výběru DR.
- **Router dead interval** udává dobu, po které se spojení se sousedem pokládá za nefunkční.
- **Designated router** a **Backup designated router** je využíváno v procesu výběru DR.
- **Neighbors** seznam směrovačů (jejich ID), od kterých směrovač (který odeslal hello zprávu) obdržel hello zprávy.



Obrázek 4. Hello paket

Po přijetí hello paketu směrovač ověří, zda se jeho ID nachází v poli Neighbor, pokud ano pokračuje se ověřením dalších parametrů. Pro úspěšně navázaní vztahu sousednosti je nutno aby směrovače měly nastavenou shodnou dobu Hello interval a Router dead interval. Navíc musí ležet ve stejné oblasti sítě. Pokud jsou splněny tyto podmínky, je směrovač umístěn do tabulky reprezentující aktuální sousedy. Rozdílnost polí Options může vést k odmítnutí navázáni vztahu.

Schéma navázání vztahu, až do stavu Two-way (navázání obousměrné komunikace) se podobá 3-fázovému handshakingu protokolu TCP/IP [10].



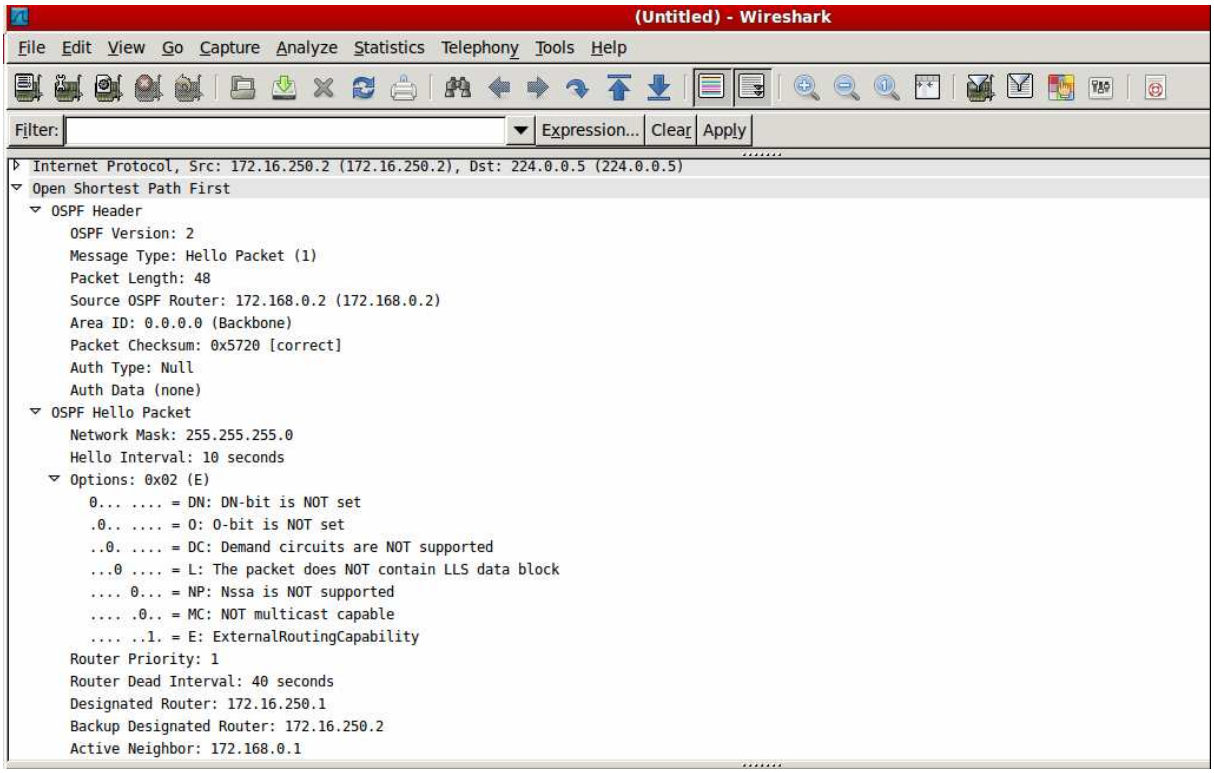
Obrázek 5.

Schéma na obrázku 5 vypadá jednoduše, ale ve skutečnosti schéma reprezentuje velké množství stavů, které proces navazování vztahu sousednosti využívá během své činnosti.

Down	Úvodní stav po inicializaci procesu OSPF. Směrovač začíná s odesíláním hello paketů.
Attempt	Stav po odeslání hello paketu. Vyskytuje se pouze u NBMA sítí.
Init	Směrovač přijal hello paket od sousedního směrovače, ale nenachází se v poli Neighbors, které se nachází na konci hello paketu.
Two-way	Úspěšné navázaní obousměrné komunikace. Směrovač se nachází v poli Neighbors přijatého hello paketu. V této fázi jsou směrovače ve vztahu sousednosti. V broadcast sítích na konci této fáze proběhne výběr DR.
Exstart	Směrovače se dohodnou na počátečních sekvenčních číslech a zvolí se master a slave směrovač. Master iniciuje další komunikaci.
Exchange	Výměna DDP.
Loading	Probíhá synchronizace databází stavu linek. Směrovač s neaktuální databází paketem LSR požádá o nová data. Druhý směrovač mu je paketem LSU odešle a čeká na potvrzení paketem LSAck.
Full	Kompletní dokončení procesu. Směrovače mezi sebou vytvoří užší vazbu - jsou přilehlé (adjacent). Sesynchronizují se databáze stavu linek. V tomto stavu se má nacházet vazba mezi směrovači při správné funkci.

Tabulka 1. Fáze navazování vztahu přilehlosti

Vztah sousednosti se udržuje aktivní také pomocí hello zpráv. Každé přijetí této zprávy vynuluje čas od přijetí poslední hello zprávy. Pokud po dobu Router dead interval směrovač neobdrží od svého souseda hello zprávu, tak se spojení s ním pokládá za nefunkční a vztah se ukončí. [2]



Obrázek 6. Ukázka hello paketu

Na 6. obrázku je možno vidět reálnou hello zprávu z komunikace dvou směrovačů. Paket byl zachycen pomocí programu WireShark. Hodnotu intervalů Hello a Router dead žádný standard nebo norma nepřikazuje, ale společnosti Cisco Systems a Juniper Networks doporučují nastavit Hello interval na 10 sekund. Obvyklá délka Router dead intervalu se volí jako čtyřnásobek Hello intervalu, nejčastěji je to tedy 40s.

Jak jsem se již dříve zmínil tak navazování vztahů pomocí hello protokolu se liší v závislostí na tom, na jakém typu sítě je vztah navazován. Nejjednodušší případ představuje síť typu point-to-point. Spojené směrovače se vždy stanou přilehlými. Komunikace probíhá na multicastové adrese 224.0.0.5 (AllSPFRouters).

V broadcast sítích by bylo příliš náročné navazovat vztahy mezi všemi směrovači na jednom segmentu sítě (příliš velký datový tok pro provoz protokolu). Proto se zvolí jeden směrovač jako pověřený (designated router - DR), v případě poruchy tohoto směrovače jeho funkci převezme záložní pověřený směrovač (backup designated router - BDR) a zvolí se nový BDR. Pověřený směrovač reprezentuje svůj segment sítě navenek a pouze on může předávat směrovací informace do ostatních segmentů. Do vztahu typu Full (dle Tabulky 1.) se dostávají pouze obecné směrovače s DR a BDR. Obecné směrovače se spolu dostanou pouze do stavu

Two-way. DR se volí na základě tzv. **Router priority**. Pomocí hello paketů si sousední směrovače vymění své priority. Ten s nejvyšší prioritou se stane DR, jako BDR se volí s druhou nejvyšší. Pokud mají prioritu stejnou, tak se DR stane směrovač s vyšším RouterID. Jako DR/BDR by se měl volit nejvýkonnější nebo nejméně zatěžovaný směrovač segmentu (volbu ovlivníme nastavením priority). Komunikace probíhá pomocí multicastových adres **AllSPFRouters** (všechny směrovače segmentu) a **AllDesignatedRouters** (pouze DR a BDR).

Jelikož sítě NBMA nepodporují broadcast, tak sousední směrovače se musí prvotně načíst z konfigurace sítě. Konfigurace je omezená pouze na ruční nastavení seznamu směrovačů, které jsou schopny stát se DR (jejich priorita se nerovná nule). Pro zjednodušení správy sítě a pro omezení provozu hello protokolu se většině směrovačů, které jsou připojeny k NBMA síti, volí priorita 0. Směrovač, který má nastavenou prioritu 0 (nutnost předchozí konfigurace), odešle všem ostatním směrovačům s prioritou 0 hello paket, jakmile se jeho rozhraní stane schopné komunikace. Následně proběhne výběr DR a BDR. Vybrané směrovače začnou následně rozesílat hello pakety všem ostatním směrovačům na síti a následuje standardní navazování vztahů (obyčejné směrovače s DR a BDR).

Jelikož se v sítích typu point-to-multipoint nevolí DR a BDR, tak úlohou hello protokolu v těchto sítích je pouze detekovat sousední OSPF směrovače. Hello zprávy se periodicky odesílají pouze na přímo připojená zařízení. [3]

Volba DR a BDR v protokolu OSPF probíhá nepreemptivně.

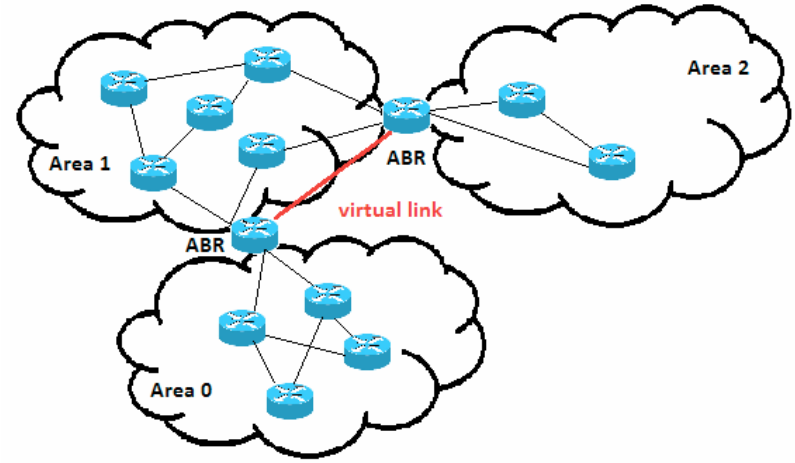
3.5. Hierarchie v OSPF, oblasti a typy oblastí

V síti, která má stabilní topologii, se šíří minimum směrovacích a řídících informací - periodicky se odesílají krátké hello pakety a co 30 minut dojde k obnovení databáze stavu linek. Nicméně, když bude síť často měnit svou topologii, bude se celou sítí šířit velké množství LSU paketů, které budou tyto změny popisovat, a jejich potvrzovací informace. Navíc po každé změně databáze stavu linek se provede nový výpočet nejkratších cest. Toto vše bude zatěžovat jak samotnou síť a její linky, tak procesor každého směrovače.

Zlepšení tohoto problému představuje možnost rozdělení sítě (autonomního systému) na oblasti. V každé oblasti je aktivní samostatná instance směrovacího procesu. Směrovače spravují databáze stavu linek oblastí, do kterých jsou připojeny jejich rozhraní. Směrovače připojené do více oblastí se nazývají hraniční směrovače (ABR - Area Border Router). Ze zbytku oblastí obdrží pouze souhrnná data. Jako oblast se označuje nějaká logická množina směrovačů a linek mezi těmito směrovači. Rozdělení sítě do oblastí umožňuje protokolu pracovat v rozlehlých sítích. [2]

Oblasti označujeme 32 bitovým číslem (area 10, někde se používá zápis značení ve formě IP adresy - area 0.0.0.128). OSPF páteřní oblast je speciální oblastí (označujeme jako area 0, area 0.0.0.0). Pomocí ní jsou spojeny všechny ostatní oblasti celého autonomního systému, obsahuje všechny ABR (výjimku tvoří ABR spojený s páteří virtuální linkou). Provoz směrující mezi libovolnými oblastmi musí procházet právě přes páteřní oblast. Nicméně, ne vždy je možno tuto podmínku dodržet. Mohou vzniknout topologie, kde je nemožné podmínku

splnit. V tomto případě můžeme vytvořit virtuální linku mezi ABR páteřní oblasti a ABR jiné oblasti, ovšem tato linka může procházet právě jednou nepáteřní oblastí. Viz obrázek 7.



Obrázek 7. Virtual link

Při směrování paketů mezi dvěma nepáteřními oblastmi se využívá páteřní oblast. Cesta mezi zařízeními z různých oblastí se může dělit na tři dílčí části: cesta na hranici oblasti, cesta páteřní sítí mezi dvěma oblastmi a cestu z hranice cílové oblasti do cílového zařízení. Směrování probíhá na každé ze tři části nezávisle na ostatních. Z tohoto pohledu můžeme na směrování mezi oblastmi nahlížet jako na topologii hvězdy, kde páteřní oblast tvoří rozbočovač a zbylé oblasti tvoří cílové destinace.

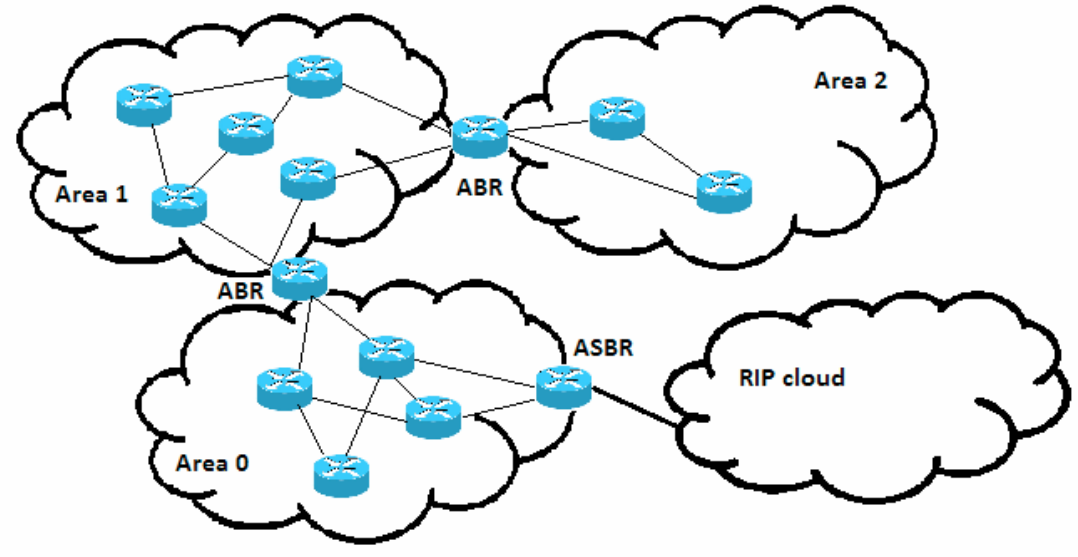
3.5.1. Spolupráce OSPF s jinými směrovacími protokoly

V reálném provozu je ovšem někdy nutno spojit sítě, které nevyužívají stejný směrovací protokol. Není možné aby na všech sítích fungoval stejný protokol. Na hranici autonomního systému (z pohledu OSPF nikoli externího směrování) můžeme nalézt směrovače, které podporují více směrovacích protokolů (spolu s OSPF). Tyto směrovače označujeme jako ASBR (hraniční směrovač autonomního systému). Úkolem ASBR je importovat směrovací informace z externí oblasti, do směrovací domény OSPF. Tento proces označujeme jako **redistribuci**. Umožňuje směrovačům určit nejlepší odchozí cestu z autonomního systému OSPF do oblasti s jiným směrovacím protokolem. Směrovací informace z jiných směrovacích protokolů a z jiných autonomních systémů jsou označované jako externí směrovací informace nebo externí cesty. Ukázka topologie využívající různé směrovací protokoly je na obrázku 8.

Problém, který se řeší při redistribuci, je nekompatibilita metrik různých protokolů (např. cena u OSPF a počet přeskoků u RIP). Při konfiguraci je nutno nastavit cenu, která se přiřadí při propagaci sítí mezi systémy (cena sítí z RIP oblasti v oblasti OSPF).

Protokol OSPF rozlišuje dva typy externích cest - Externí 1 (E1) a Externí 2 (E2). U cesty typu E2 je její cena dána pouze cenou, se kterou byla tato cesta do OSPF distribuována,

není tedy závislá na cestách ve vnitřku OSPF autonomního systému. Oproti tomu je výsledná cesta u cesty typu E1 dána součtem původní ceny, kterou cesta dostala na vstupu do OSPF domény a cenami dílčích cest uvnitř OSPF autonomního systému. Častěji se využívají cesty typu E2, využití cesty typu E1 je vhodné pokud na hranici oblastí s různými směrovacími protokoly leží více směrovačů - existence více možných cest pryč z OSPF autonomního systému. [2]



Obrázek 8. ASBR

V případě, že existuje více ekvivalentních cest k ASBR na okraji OSPF autonomního systému, tak se uplatňují dva základní pravidla pro výběr cesty. Tyto pravidla se uplatňují když ASBR je dosažitelné přes různé oblasti.

- Cesty, které procházejí přes nepáteřní oblast, jsou vždy preferovány.
- Ostatní typy cest, cesty skrz páteřní oblast a mezi-oblastní cesty mají stejnou úroveň upřednostňování.

3.5.2. Stub area

Základně se oblasti dělí na dva typy oblastí: **stub** a **tranzitní**. Toto dělení se určuje na základě typu provozu procházejícího přes danou oblast. Přes tranzitní oblast provoz pouze protéká do jiné oblasti (nejtypičtějším představitelem tranzitní oblasti je páteřní oblast - **area** 0, ale může to být i jiná oblast, pokud je v ní umístěn hraniční směrovač autonomního systému), ve stub oblasti provoz začíná nebo končí, ale nikdy jí neprotéká.

Stub oblast přináší oblasti řadu výhod. Pokud oblast takto nakonfigurujeme, tak se do ní nebudou propagovat externí cesty, ale pouze ty, které nalezl OSPF proces v daném autonomním systému. Směrování do externích sítí se řeší pomocí tzv. explicitní cesty. Tuto

cestu do této oblasti propaguje ASBR pomocí LSA typu 3, které se na ABR stub oblasti transformuje na LSA typu 1, komunikace stub oblastí (s externími sítěmi) tedy musí téct přes ASBR. Přínosem stub oblastí je to, že se sníží velikost databází stavu linek (klesne zatížení paměti i procesoru směrovače).

Stub oblast musí splňovat tyto podmínky: v oblasti se nesmí nacházet virtuální linka (ani touto oblastí procházet), v oblasti nesmí být přítomen ASBR, všechny její směrovače musí být nakonfigurovány jako stub a oblast nesmí být páteřní oblastí (area 0). Výhodou je pokud z oblasti vede ven pouze jedna cesta, ale není to nutná podmínka. V případě více cest, které vedou z oblasti ven není zaručen přenos dat do externích sítí nejkratší cestou. [2]

3.5.3. Totally stubby area

Totally stubby oblast rozvíjí myšlenku a koncepci stub oblastí ještě více. Pokud z oblasti vede pouze jediná cesta do jiné oblasti, tak se do totally stubby oblasti bude propagovat pouze explicitní cesta. Nebudou se tedy do ní propagovat cesty z jiných oblastí stejného AS. Ve směrovacích tabulkách směrovačů v této oblasti nalezneme pouze záznamy směrující v dané oblasti a explicitní cestu. Tento typ oblasti nevyužívá LSA typu 4 a 5. Tento typ konfigurace není standardní součástí protokolu OSPF, ale je proprietární implementací společnosti Cisco systems. [2]

3.5.4. Not so stubby area

Tento typ oblasti se od stub oblasti příliš neliší. V této oblasti je povolená přítomnost ASBR. Tento typ oblasti vznikl z důvodu potřeby redistribuce směrovacích dat jiného protokolu v oblasti, která je vhodným kandidátem na vytvoření stub oblasti. Hraniční směrovač oblasti se tím pádem stává ABSR, který není v klasické stub oblasti povolen. V této oblasti se propagují LSA typu 7, které se na ABR převádějí na LSA typu 5. [2]

3.6. Link State Advertisement

Důležitým mechanismem protokolu OSPF je výměna směrovacích informací. Tento proces zabezpečuje distribuci, aktualizaci a synchronizaci databází stavu linek na jednotlivých směrovačích.

Data jsou předávaná pomocí tří typů zpráv: Link state request, link state update a link state acknowledge. Všechny typy zpráv využívají záhlaví klasického OSPF paketu a liší se pouze v datové části.

Při prvním navazování vztahu přilehlosti si směrovače vymění DDP, které popisují aktuální stav databáze (sekvenční číslo a jednotlivé hlavičky LSA). Následně se databáze synchronizují.

Pokud jsou již směrovače přilehlé, tak se v případě aktualizace LSDB (připojení nového rozhraní, změna sítě na rozhraní, výpadek na lince) rozešle LSU paket a pomocí procesu zaplavení dojde k distribuci nových informací.

Součástí mechanismu zaplavování není pouze funkce pro přenos, ale taky funkce pro zajištění spolehlivosti přenosu a změn databáze:

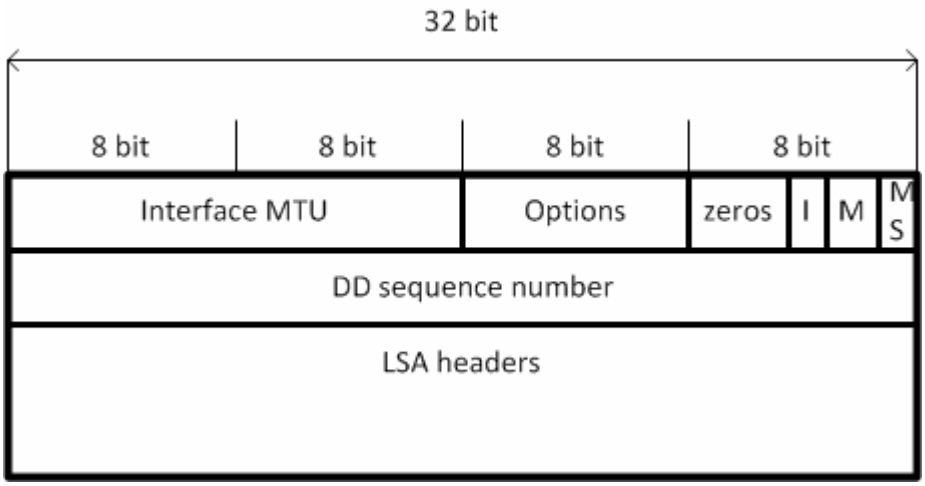
- **Zapouzdření** přenášených informací při transportu mezi směrovači.
- **Spolehlivost aktualizace databáze**: sekvenční čísla jednotlivých LSA i celých databází, ukládání stáří jednotlivých LSA a kontrolní součty.
- Potvrzování zpráv, které byly přijaty pomocí mechanismu zaplavování.
- Soubor pravidel spravující informace o tom, které typy zpráv může směrovač přijmout, zpracovat a případně odeslat do dalších segmentů sítě. Informace se kterými směrovač neumí pracovat, jsou zahozen, nesmí se propagovat dál do sítě.

3.6.1. Database Description Packet

OSPF zpráva typu 2. Využívá se v procesu výměny databází mezi sousedními směrovači. Je využíván pro popis současného stavu LSDB. Paket využívá standardní OSPF hlavičku, za kterou umístí své záhlaví.

Jako první se v hlavičce DDP paketu nachází údaj o MTU rozhraní směrovače, který jej odeslal. Tento údaj je velice důležitý pro přenos mezi směrovači. Pokud směrovač, který přijal DDP, neumí zpracovávat MTU o dané velikosti, musí byt DDP zahozen. Pokud se DDP posílají pomocí virtuální linky, mělo by pole být nastaveno na 0.

DD sequence number se využívá pro očíslování částí LSDB. Počáteční hodnota na začátku přenosu (indikován I bitem) by měla být unikátní. Následně se číslo inkrementuje dokud není odeslán popis celé databáze. Schéma paketu je znázorněno na obrázku 9.



Obrázek 9. - DDP paket.

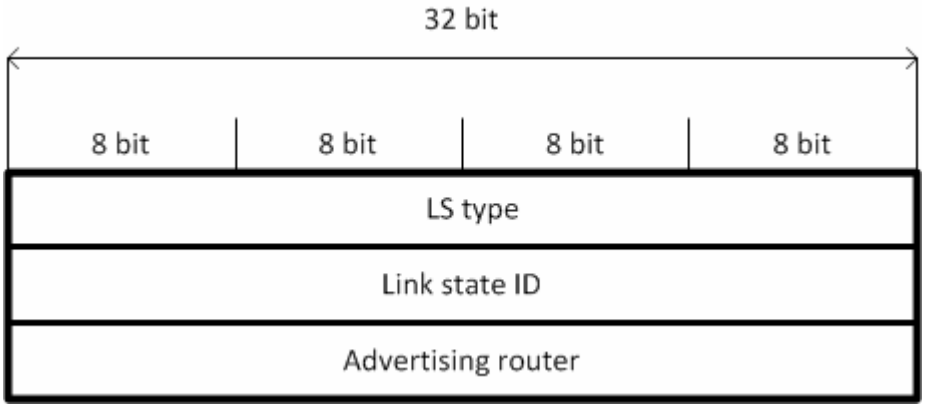
Speciální pole je v tomto paketu umístěno za polem options. Prvních 5 bitů se nastavuje automaticky na hodnotu 0 a nejsou nijak využívány (jsou rezervovány pro budoucí použití) a za nimi se nacházejí tři řídící bity:

- **I bit** - Init bit. Pokud je jeho hodnota rovna 1, tak je paket první ze sekvence DDP.
- **M bit** - More bit. je nastaven na hodnotu 1, pokud se za paketem nacházejí další DDP.
- **MS bit** - Master/slave bit. Směrovače při výměně databází navážou Master/slave relaci. Pokud je hodnota tohoto bitu nastavená na 1, tak směrovač je při výměně databází v roli Master.

Za hlavičkou DDP se již nachází samotná data popisující databázi směrovače.

3.6.2. LSR, LSU, LSAck

Po zpracování DDP může směrovač zjistit, že část jeho databáze již není aktuální. Pokud nastane tato situace, tak pomocí paketu Link State Request (OSPF paket typ 3) si směrovač vyžádá tu část databáze, kterou má zastaralou. Struktura zprávy je ukázána na obrázku 10.

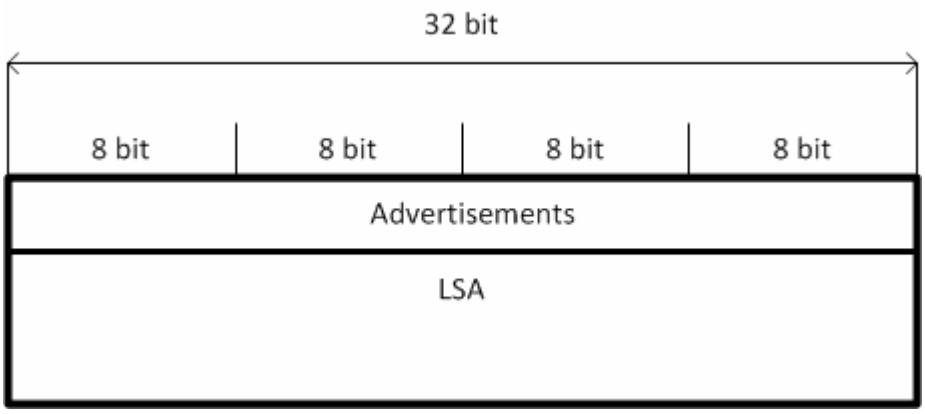


Obrázek 10. - LSR paket.

Každé LSA je specifikováno svým typem (LS type), svým jednoznačným identifikátorem (LS ID) a propagujícím směrovačem. Tato trojice jednoznačně identifikuje LSA, nikoli jeho konkrétní instanci. LSR pakety jsou chápány jako vyžádání nejnovější instance LSA, nikoli konkrétní instance.

Po vyžádání konkrétních LSA odpoví směrovač odesláním vyžádaných LSA. Vyžádaná LSA odesílá paketem LSU (OSPF paket typ 4). Tento typ paketu zabezpečuje zaplavování OSPF systému směrovacími informacemi (LSA). V poli Advertisements se oznamuje počet LSA odesílaných v daném paketu. Za tímto polem již následují samotná LSA

data - každý záznam obsahuje 20 bytovou hlavičku a samotná směrovací data. Paket je znázorněn na obrázku 11.



Obrázek 11. - LSU paket.

Aby byl mechanismus zaplavování spolehlivý, musí se potvrdit jejich přijetí druhým směrovačem. Potvrzení přijatých dat se provádí prostřednictvím odeslání LSAck (OSPF paket typ 5) paketu. Paket má velice jednoduchou strukturu, obdobnou jako má paket LSU. Za standardní OSPF hlavičkou se nachází pouze uložené hlavičky přijatých LSA, nikoli celá LSA jako u LSU paketu. Tento princip potvrzování je jednoduchý a nezatěžuje výkon směrovače - pouze se zkopíruje část přijatých dat do paketu.

3.6.3. Typy LSA

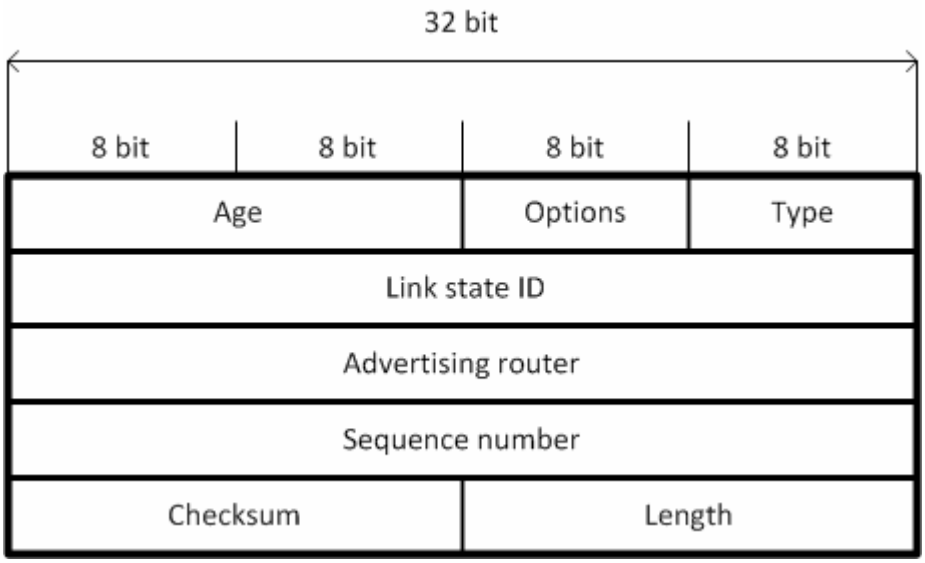
Databáze stavu linek se skládá z několika různých typů záznamů. Každý typ záznamů se využívá pro popis jiného typu cesty (cesta uvnitř oblasti, cesta mimo oblast...). Ovšem všechny typy záznamů využívají stejnou hlavičku. Pomocí LSA se popisuje celá topologie sítě, směrovače si vyměňují popisy sítí, které jsou k nim přímo připojeny.

V hlavičce LSA se nachází informace pro správné zaplavování. Jako první se v hlavičce nachází pole Age, které vyjadřuje stáří daného LSA v sekundách. Může nabývat hodnot z intervalu 0 až 3600. Pokud LSA dosáhne stáří 3600 sekund (MaxAge), je označeno za neplatné. Proto se musí LSA v době menší než MaxAge periodicky obnovovat (klasicky MaxAge / 2). Přenáší se jako neznaménkový datový typ integer.

Pole Options je shodné se stejnojmennými poli v jiných typech paketů. Položka Type specifikuje typ konkrétního LSA. Pomocí pole se identifikuje směrovač, který LSA odeslal (přenáší se jeho RouterID).

Sequence number je 32 bitové číslo. OSPF využívá lineární sekvenční číslo. Při prvním odeslání LSA směrovač nastaví hodnotu na 0x80000001, což je OSPF konstanta InitialSequenceNumber. Následně se tato hodnota inkrementuje při každém dílčím přeposlání na každém směrovači. Maximální hodnota, které může toto číslo dosáhnout je 0x7fffffff. Pokud je nová instance LSA připravená k odeslání, ale její současná verze má maximální hodnotu sekvenčního čísla, tak tato verze musí být z databáze odstraněná umělým zestárnutím. Odeslání

pak pokračuje až všechny přilehlé směrovače potvrdí odstranění staré verze. Obrázek 12 reprezentuje strukturu LSA hlavičky. [1]



Obrázek 12.

LinkStateID je 32 bitové pole přenášející IP adresu, která identifikuje dané LSA. Zdroj této IP adresy se liší u každého typu LSA. Jednotlivé typy LSA a jejich LinkStateID:

- **LSA typ 1 - Router LSA -** Popisuje přímo připojené sítě k rozhraní směrovače připojené do stejné oblasti. Šíří se pouze do oblasti, ve které leží připojená síť. LSID je RouterID směrovače, který LSA propaguje.
- **LSA typ 2 - Network LSA -** Pomocí tohoto typu LSA DR na broadcast segmentu sítě oznamuje, které směrovače jsou na daném segmentu připojeny. Šíří se pouze v oblasti do které náleží broadcast segment. Jako LSID se využívá IP adresa rozhraní DR do segmentu sítě.
- **LSA typ 3 - Network Summary LSA -** ABR tímto typem propaguje sumarizované adresy ze své oblasti do oblasti jiné. Sumarizace odstraňuje detailní popis topologie oblasti (šíří se pouze prefix a maska). LSID je shodné s adresou cílové sítě.
- **LSA typ 4 - ASBR-Summary LSA -** Jelikož LSA typu 5 je propagováno do všech oblastí sítě, tak v některých částech sítě mohou chybět informace jak dosáhnout ABSR. Při této situaci se využívá LSA typ 4, které propaguje cestu k ASBR do všech oblastí. LSID je RouterID popisovaného ASBR.
- **LSA typ 5 - AS-External LSA -** Využívá se při redistribuci směrovacích informací z jiného směrovacího procesu. Propagují se do všech oblastí. LSID se získává z adresy externí sítě (cílová adresa cest z OSPF domény).
- **LSA typ 6 - Group membership LSA -** Navrženo pro multicast OSPF (MOSPF). Od nástupu OSPFv3 se nevyužívá. Jako LSID se nastaví adresa cílové multicastové skupiny.

- **LSA typ 7 - NSSA LSA -** Směrovače v NSSA nepřijímají externí LSA z ABR, ale můžou odesílat informace o externích cestách. Pomocí tohoto typu LSA oznámí ABR tyto externí cesty a ABR je přeloží na LSA typ 5, které následně vypropaguje do zbytku sítě. LSID je shodné s adresou cílové sítě.
- **LSA typ 8 - External Atributes LSA -** původně navržen pro tranzitní autonomní systémy, kde OSPFv2 nahrazuje interní BGP. Typ 5 měl přenášet BGP cesty a typ 8 BGP atributy. V praktickém využití se neuchytilo. Většina směrovačů jej nepodporuje. LSID jsou zakódované BGP atributy.
- **LSA typ 9, 10, 11 -** Místo LSID se nastavuje 8 bitů jako Opaque type a 24 bitů jako Opaque ID.

3.6.4. Výměna LSA

Aby si směrovače mohly vzájemně posílat směrovací (LSA) informace, musí se nejprve stát přilehlými. Úvodní synchronizace databází proběhne ve dvou fázích: popis databáze pomocí DDP a samotné odeslání LSA.

Nejjednodušší způsob komunikace nastává na linkách typu point-to-point. Dva směrovače na lince spolu komunikují prostřednictvím multicastové adresy 224.0.0.5.

Na síti typu broadcast s N směrovači by bylo navázáno $N*(N-1)/2$ vztahů přilehlosti. Tento stav by vedl k velkému množství LSU a LSAck zpráv. Proto se volí DR. Všechny směrovače si pak synchronizují databázi pouze s DR. Směrovač, který zaznamená změnu v topologii sítě, vygeneruje LSU paket, který odešle na DR a ten rozešle tento LSU paket všem ostatním směrovačům na síťovém segmentu. Tento princip ušetří značné množství provozu (je navázáno pouze N vztahů přilehlosti), ale je taky náchylný na výpadek DR (v případě výpadku DR by topologické databáze byly nesynchronizovány). Proto si směrovače synchronizují databáze také s BDR, který ihned po detekci výpadku DR převezme jeho funkci. Pro tento případ se naváže $2*N-1$ vztahů přilehlosti.

Synchronizace databází v NBMA sítích probíhá obdobně jako v broadcast sítích. Zvolí se DR a BDR, se kterým si všechny ostatní směrovače synchronizují své databáze. Zaplavování probíhá taktéž prostřednictvím DR a BDR (příchozí LSU rozešle do celého segmentu sítě). Jediný rozdíl spočívá v metodě rozesílání LSU paketů. V NBMA sítích se zprávy neodesílají pomocí multicastových zpráv, ale jednotlivě pomocí unicastových na každý směrovač odděleně.

V sítích point-to-multipoint se nevolí DR a BDR. Přilehlými směrovači se stávají všechny přímo připojené směrovače.

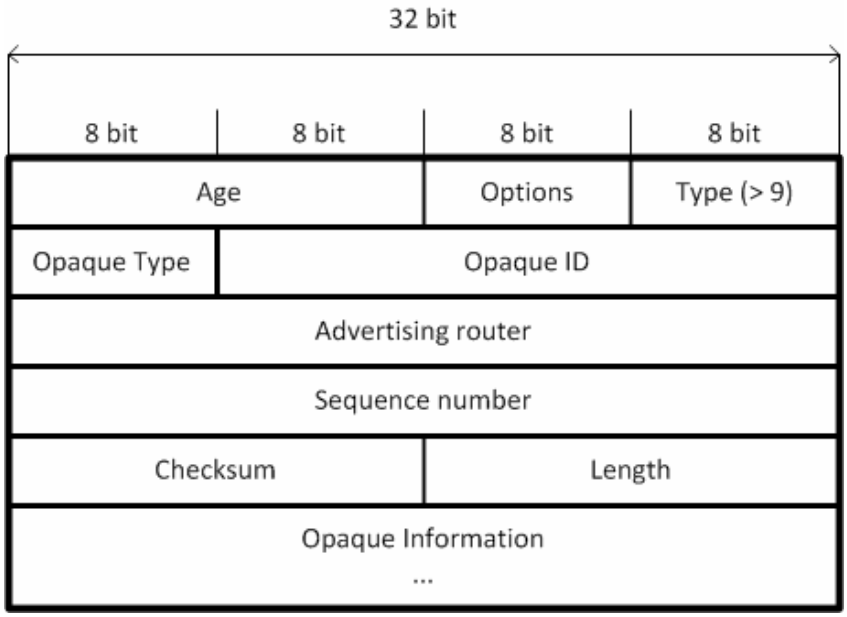
Omezení při zaplavování sítě LSA zprávami jsou nutná pro správnou funkci distribuce a zajišťují aby nedošlo k zahlcení sítě. Po přijetí LSA musí směrovač rozeznat, kterými rozhraními má informaci distribuovat dál do sítě. Na rozhraní, ze kterého směrovač LSA obdržel, smí odeslat paket LSU pouze v případě, že je použitá síť typu broadcast nebo NBMA a směrovač je zároveň vybrán jako DR. Dalším omezením je oblast, do které náleží rozhraní, které přijalo LSA. Pokud je LSA omezeno pouze na jednu oblast, tak se odesílá pouze na

rozhraní, která náleží do stejné oblasti, ze které směrovač LSA obdržel. Při dodržení těchto omezení bude LSA mechanismus pracovat správně a nebude přetěžovat síť nadbytečným šířením zpráv. [3]

3.6.5. Opaque LSA

Protokol OSPF poskytuje dodatečnou funkčnost pro další typy LSA zpráv, které nejsou nutné pro správnou funkčnost protokolu - nejsou součástí protokolu RFC2328 (standardu OSPF, definuje je RFC2370). Tyto zprávy mohou být nicméně využity protokolem OSPF nebo jakoukoliv jinou aplikací, která si pouze prostřednictvím OSPF posílá data - využívá distribuční schopnosti protokolu. Implementace opaque LSA poskytuje aplikaci rozhraní pro enkapsulaci specifických dat do paque LSA typu. Dále umožňuje aplikaci přijímat a odesílat data a kontrolovat jejich správnost (i v případě změny topologie sítě).

Opaque LSA jsou typu 9, 10 a 11. Skládají se ze standardní LSA hlavičky, která je následovaná speciálním 32 bitovým informačním polem. Pro distribuci jsou využity standardní OSPF LSA distribuční mechanismy. Směrovače, které nepodporují Opaque LSA, musí tyto pakety zahazovat a nedistribuovat je dál.



Obrázek 13.

Na obrázku 13 je vidět, že hlavička Opaque LSA je shodná s klasickou LSA hlavičkou. Jediný rozdíl je v poli LinkStateID, které nahradily pole OpaqueType/ID. Typ LSA je určen polem Type. Typ u Opaque LSA rozsah distribuce v topologii sítě [8]:

- **Typ 9 - link local scope -** LSA se nesmí propagovat dál, než za hranici lokální sítě.

- **Typ 10 - area local scope -** LSA tohoto typu se nesmí distribuovat mimo hranice oblasti, ve které bylo vytvořeno.
- **Typ 11 - AS scope -** Toto LSA je propagováno v celém OSPF autonomním systému. Nesmí se ale propagovat do stub oblastí a NSSA.

Pole OpaqueID má stejnou funkci jako LSID, jednoznačně identifikuje Opaque LSA. OpaqueType identifikuje samotný typ přenášených dat v LSA paketu. Tyto typy spravuje IANA. Rozšíření protokolu OSPF o nové typy musí být akceptováno OSPF working group. Typy 0 až 127 jsou alokovány se souhlasem IETF a typy 128 až 255 jsou rezervovány pro soukromé nebo experimentální potřeby. [6]

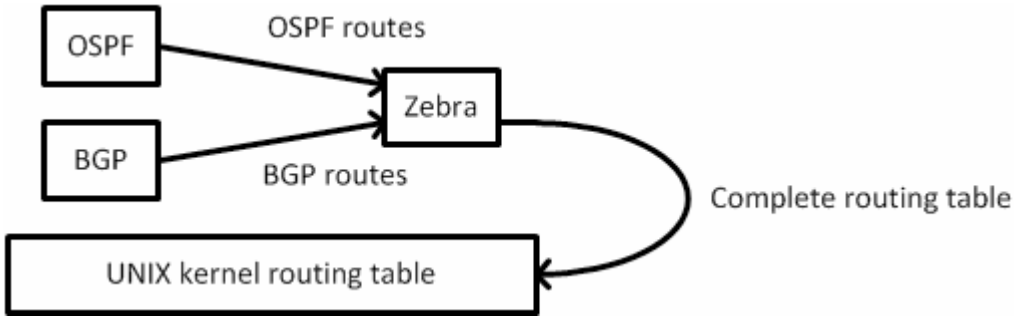
4. Quagga

Quagga je směrovací software poskytující směrovací služby založené na protokolech rodiny TCP/IP, jako jsou: RIP, OSPF, BGP, ale také protokol IS-IS, který se mezi protokoly rodiny TCP/IP neřadí. Rovněž podporuje speciální službu BGP Route Reflector a chování Route serveru. Poskytuje možnost směrování i pro IPv6. Quagga je distribuovaná pod licencí GNU General Public License. Celá aplikace je napsaná v jazyce C. [5]

4.1. Architektura

Quagga nepracuje pouze v jediném procesu, který poskytuje veškeré směrovací funkce, ale je rozdělená do několika samostatných démonů. Tento přístup umožňuje provozování jednotlivých démonů na různých zařízeních (komunikují spolu pomocí socketů) nebo provoz více procesů stejného protokolu na jedné stanici. Provoz na různých stanicích zvyšuje stabilitu systému. Pokud zhavaruje jeden systém (např. provozující OSPF), tak Quagga jako celek pracuje dál a může využít cesty jiného protokolu.

Na obrázku 14 je znázorněná spolupráce jednotlivých dílčích části Quaggy. Jednotlivé směrovací protokoly pracují nezávisle na sobě a následně předají své směrovací tabulky řídící části Zebra, která sestaví výslednou směrovací tabulku (na základě administrativních vzdáleností) a tu následně předá jádru operačního systému, které pak realizuje samotné směrování. Schéma předávání cest je ukázáno na obrázku 14.



Obrázek 14.

Tento více procesový přístup přináší modularitu, rozšiřitelnost a snadnější údržbu celého systému. Toto řešení taktéž poskytuje oddělené konfigurační soubory jednotlivých démonů a samostatné terminálové rozhraní pro konfiguraci dílčích částí (protokolů) systému.

Konfigurace jednotlivých protokolů se provádí pomocí konfiguračních souborů (např. nejčastěji umístěny ve složce */usr/local/etc* a mají název ve tvaru *název_démonu.conf*). Statické cesty se konfigurují v *zebra.conf*, ale sítě pro protokol OSPF v souboru *ospfd.conf,* což může být nepraktické, proto můžeme jednotlivé konfigurace provádět pomocí virtuálního terminálu, který se připojí k jednotlivým konfiguračním rozhraním dílčích démonů.

4.2. Podporované platformy

V současné verzi Quagga (verze 0.99.22) podporuje platformy GNU/Linux a BSD. Ovšem migrace na jinou platformu není obtížná, platformě závislý kód se nachází pouze v démonu *zebra*. Protokolové démony jsou platformě nezávislé.

Seznam oficiálně podporovaných platforem, Quagga může běžet korektně i na jiných platformách, případně na jiných platformách pouze s částečnou funkčností (zdroj [5]):

- GNU/Linux
- FreeBSD
- NetBSD
- OpenBSD

Quaggu může ovlivnit i výběr kompilátoru a knihoven jazyka C. Otestovány byly tyto kompilátory jazyka C:

- GNU GCC
- LLVM clang
- Intel ICC

4.3. Instalace

Konfigurace, kompilace a instalace Quaggy je velice jednoduchá. Celá proběhne po spuštění tří příkazů: *configure, make* a *make install*. Tyto příkazy musíme zadat v kořenové složce Quaggy. Nejdůležitějším příkazem je patrně konfigurační příkaz configure, který popíši v další části. Příkazem make se provede kompilace zdrojových kódů a make install zkompilované balíčky a knihovny nainstaluje do systému a nastaví všechny potřebné hodnoty v konfiguraci systému (např. jméno démonů pro připojení pomocí telnet).

Po samotné instalaci programu je pro správné fungování nutno zadat příkaz ldconfig (který vytvoří odkazy na nejnovější sdílené knihovny v systému). Také musíme vytvořit skupinu uživatelů Quagga a do ní přidat uživatele Quagga. Tento uživatelský účet bude Quagga implicitně využívat pro své spuštění.

4.3.1. Konfigurace instalace

Tato část je pro správné provozování velmi důležitá. Při příkazu *configure* se vytváří soubor *config.h*, který při následné kompilaci ovlivňuje kompilované části kódu. Tento příkaz rovněž vytváří makefile jednotlivých částí projektu (démonů) a nastavuje hodnoty systémově

závislých konstan. Také ovlivňuje funkčnost programu, který se bude instalovat. Pro funkci OpaqueLSA je nutno konfiguraci spustit následujícím příkazem:

```
./configure --enable-opaque-lsa
```

Užitečné je při konfiguraci zakázat kompilaci a instalaci nevyužívaných protokolů (--disable-ripd, --disable-bgpd, --disable-ospf6d), tímto zakázáním zkrátíme čas kompilace a snížíme paměťové nároky kladené na systém.

Pomocí přepínače --prefix=nějaký_prefix se určí prefix složek, do kterých se ukládají pracovní soubory - do prefix/etc se ukládájí konfigurační soubory a do složky prefix/sbin se ukládají spouštěcí soubory (explicitní hodnota prefix = /usr/local).

4.4. Srovnání Quaggy se směrovači Cisco

	Quagga v.0.9.22	Cisco 2811	Cisco 3750 L3
Statické směrování	ANO	ANO	ANO
RIPv2	ANO	ANO	ANO
OSPFv2	ANO	ANO	ANO
BGPv4	ANO	ANO	ANO
NAT	NE / UNIX	ANO	NE
VRRP	NE / UNIX	ANO	NE
VPN IPSec	NE / UNIX	ANO	ANO
VPN SSL	NE / UNIX	NE	NE
DHCP server	NE / UNIX	ANO	ANO
SNMP	ANO	ANO	ANO

Tabulka 2.

Z hlediska poskytovaných funkcí Quagga oproti fyzickým směšovačům nezaostává (musíme ovšem vhodně kombinovat funkce s funkcemi OS), některé funkce je ovšem nutno kombinovat s poskytovanou funkčností operačního systému. Její nevýhodou je absence specializovaného hardwaru, absence technické podpory. Její výhodou je možnost úpravy kódu směrovacích protokolů (můžeme si protokol přizpůsobit našim požadavkům) a pořizovací cena.

5. Implementace

Ještě před samotnou implementací jsem se musel důkladně seznámit se strukturou kódu Quaggy. Celý projekt je rozdělen na dílčí části pomocí složek. Každá složka nese název ve tvaru *jméno_protokolud*, kde písmeno "d" označuje démon. V kořenové složce projektu se nachází konfigurační soubor. Samotný kód je pak rozdělen na hlavičkové soubory (definují se v nich struktury, konstanty, těla datových struktur a hlavičky metod) a soubory, ve kterých se již implementuje samotná funkčnost, kód je psán s využitím metod a funkcí.

5.1. Současný stav

Jak jsem se již zmínil v úvodu, existuje několik volně šiřitelných implementací směrovacího protokolu OSPF, ale žádná neposkytuje ve svém základu možnost transportu libovolných uživatelských dat. Je tedy nutno, na základě speciálních požadavků, provést analýzu možných řešení a vybrat nejoptimálnější. Na základě vybraného řešení pak provedu samotnou implementaci požadavků.

5.2. Požadavky na program

Aplikace by měla splňovat následující požadavky:

- **Spolehlivost –** distribuce našich položek by měla probíhat automaticky a spolehlivě mezi všemi směrovači v oblasti sítě.
- **Přenášet data ve formátu TLV –** Type identifikující typ přenášených dat, Length určující délku přenášených dat a Value (hodnota), např. ve formátu IP adresa zařízení a MAC adresa zařízení.
- **Data získávat nezávisle na uživateli –** činnost mechanismu získávání externích dat má probíhat nezávisle na uživateli protokolu.
- **Rozšíření existující implementace OSPF –** aplikace by měla být založená existující implementaci protokolu OSPF, která má ověřenou základní funkčnost.
- **Výpis přenášených TLV položek -** aplikace by měla poskytovat možnost přijatá data vypsat do souboru a do konzole.

5.3. Analýza možností řešení

Po prostudování RFC dokumentu protokolu OSPFv2 jsem zjistil, že existuje několik variant řešení problému. Některé z možností zde popíši. Všechny možnosti by měly společnou pouze datovou strukturu, reprezentující jednotlivé datové záznamy.

První možností řešení by bylo navrhnout a implementovat vlastní typ LSA. Pro toto řešení by bylo nutné navrhnout zaplavovací pravidla a politiky. Následně by bylo nutno definovat i logiku zpracovávající přijatá data. Tento navržený mechanismus by musel být robustní a musel by se velice důkladně testovat. Jeho výhoda by byla v tom, že veškeré mechanismy distribuování by fungovaly dle našich požadavků a pro správce by byl transparentní.

Další variantou implementace by bylo navržení vlastního typu Opaque LSA. Tento postup by využíval jíž existující mechanismy distribuce a zpracování LSA položek. Jeho výhodou by bylo využití existujících a funkčních mechanismů, které by vyžadovaly méně testování a analýzy provozu.

Poslední způsob implementace by nespočíval přímo v úpravě směrovacího protokolu Quaggy, ale využíval by funkčnosti Quaggy, která poskytuje externí rozhraní pro připojení klienta pomocí socketů. Takto připojený klient si může zaregistrovat vlastní LSA typ a ovládat mechanismy OSPF démona externě. Toto řešení přináší výhodu v tom, že nijak nezasahuje do kódu protokolu a plně využívá jeho poskytované rozhraní. Ovšem takovéto řešení by nesplňovalo podmínky zadání, které vyžaduje úpravu implementace protokolu OSPF.

5.4. Výběr vhodné metody řešení

Pro implementaci jsem se rozhodl vybrat možnost návrhu vlastního typu Opaque LSA. Toto řešení přináší oproti ostatním možnostem nejvíc výhod. Takto upravený protokol by pořád splňoval vlastnosti RFC, pro protokol OSPFv2 s podporou Opaque LSA je to RFC2370.

Toto řešení je kombinací přístupu využití poskytovaných funkčností implementace protokolu a navržení úplně nového typu LSA. Pro mnou definované a navržené Opaque LSA využiji všechny distribuční mechanismy, které protokol poskytuje.

Poslední výhodou by bylo, že kdyby se v síti vyskytovaly i jiné směrovače než Quagga, upravená pro naše potřeby, tak by se přijaté LSA alespoň distribuovaly dál do sítě a nebyly by zahazované.

5.5. Implementovaná funkčnost

Rozšíření směrovacího protokolu o požadovanou funkčnost lze rozdělit do několika dílčích kroků. Jako první si navrhnu datovou strukturu tak, aby splňovala požadavky na přenášená data. Jako další budu muset vyřešit vytváření jednotlivých LSA záznamů. Jelikož se jednotlivé záznamy musí periodicky obnovovat, budu muset vyřešit i jejich obnovu. Když se mi podaří tyto dílčí problémy úspěšně vyřešit, tak se zaměřím na napojení aplikace na externí zdroj dat, v tomto případě to bude ARP cache operačního systému. Na konec se zaměřím na vypisování jednotlivých LSA na konzole a souboru.

5.5.1. Definice dat a jejich struktury

Jako první krok je nutno nadefinovat jméno a číslo typu mnou definovaného LSA (v souboru *ospfd/ospf_opaque.h*). Tuto hodnotu budu využívat ve všech úpravách, které budu provádět.

```
#define OPAQUE_TYPE_TLV_LSA                    230
```

Hodnota 230 nemá specifický význam. V Quaddze již byly definovány hodnoty 224 a 225, proto jsem se rozhodl zvolit nejbližší vyšší hodnotu, zaokrouhlenou na desítky.

Strukturu pro LSA, která splňuje požadavky jsem navrhl následovně (její definice se nachází v souboru *ospfd/ospf_lsa.h)* :

```
struct tlv_lsa
{
    struct lsa_header header;

    struct in_addr IP;
    char MAC[18];
    u_char flags[2]; /*for future use*/
};
```

Jako první se ve struktuře nachází LSA hlavička, která je již v aplikaci definovaná, proto jsem se ji rozhodl využít a nedefinovat vlastní strukturu. Pro uložení IP adresy jsem se rozhodl využít datovou strukturu *in_addr*, pro tuto strukturu jsou již implementovány metody pro převod IP adresy ze zápisu xxx.xxx.xxx.xxx na 32 bitovou podobu. MAC adresu jsem se rozhodl ponechat ve tvaru prostého textu, neboli pole o délce 18-ti znaků.

Jelikož délka přenášeného záznamu (v bytech) musí být dělitelná číslem 4 beze zbytku a součet délky IP a MAC adresy je roven 22 bytům, tak jsem musel přidat dva byty, které doplnily délku do 24 bytů. Tyto 2 byty jsem v mých úpravách nijak nevyužil, nicméně v případě, že by někdo chtěl mou verzi Quaggy upravit pro své potřeby, může tyto dva byty využít jako příznaková pole nebo je využít pro přenos nějaké další dílčí řídící informace.

5.5.2. Vytvoření nového LSA a jeho vložení do databáze

Ještě než je možno uživatelské Opaque LSA vytvořit, je nutno v procesu protokolu OSPF inicializovat struktury nezbytné pro správnou funkčnost Opaque LSA. První funkce v aktuální oblasti (v té, ve které chceme využívat LSA typu 10) inicializuje struktury a čítače pro korektní funkčnost LSA 10. Tato inicializace musí proběhnout na každém směrovači zvlášť, musí se takto inicializovat všechny OSPF procesy.

```
int a = ospf_opaque_type10_lsa_init(area);
ospf_opaque_init ();
```

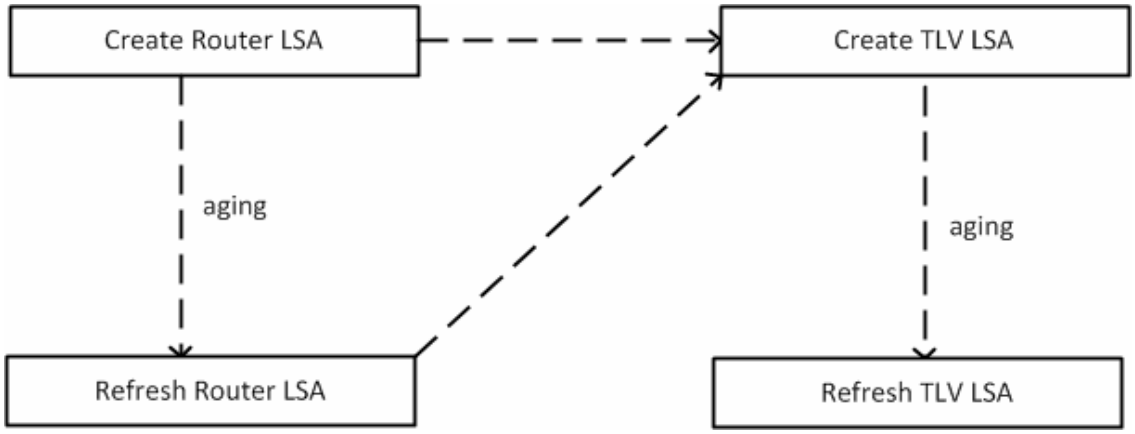
Druhá metoda inicializuje možnost využívání speciálních funkčních tabulek, ve kterých se jednotlivým LSA přiřazují metody, které tento typ LSA bude využívat. Přesněji dvojici LSA typ - Opaque Typ, každá takováto dvojice má přiřazené své metody. Je to z důvodu využívání více Opaque typů pro LSA typ 10. Jak sem se zmínil v teoretické části, tak u Opaque LSA typ LSA určuje to, v jakém rozsahu se bude dané LSA distribuovat (lokální link, oblast, celý autonomní systém OSPF).

Po těchto úvodních inicializacích jsem musel zaregistrovat funkce, které se budou využívat pro obsluhu mnou definovaného typu Opaque LSA.

```
ospf_register_opaque_functab(          Registrovaná funkce
OSPF_OPAQUE_AREA_LSA,                  LSA type  (konstanta)
  OPAQUE_TYPE_TLV_LSA,                 Opaque type  (konstanta)
                    NULL,              int (* new_if_hook)
                    NULL,              int (* del_if_hook)
                    NULL,              void (* ism_change_hook)
                    NULL,              void (* nsm_change_hook)
                    NULL,              void (* config_write_router)
                    NULL,              void (* config_write_if)
                    NULL,              void (* config_write_debug)
          tlv_lsa_detail_info,         void (* show_opaque_info)
                    NULL,              int  (* lsa_originator)
       ospf_tlv_lsa_refresh,           struct ospf_lsa *(* lsa_refresher )
                    NULL,              int (* new_lsa_hook)
                    NULL);             int (* del_lsa_hook)
```

Pro mou implementaci jsem potřeboval funkci pro obnovení (refresh) LSA záznamů. A metodu pro detailní výpis informací, které jsou uložené v konkrétním LSA záznamu. Veškerou funkčnost jsem připojil na automatické události Quaggy.

Vytváření mých Opaque LSA jsem připojil na dvě události, které se váží k Router LSA. Jedná se o vytvoření a obnovu (refresh) Router LSA. Tyto události probíhají zcela automaticky a spolehlivě – Router LSA se vytvoří vždy, při nastavení IP adresy na rozhraní směrovače. Pokud by na rozhraních nebyly nastaveny adresy, směrovací funkce by neměla smysl, tudíž by nebyla potřeba evidovat MAC adresy připojených zařízení, jelikož by směrovačem netekl žádný provoz. Mnou definované Opaque LSA v těchto událostech vytvářím pouze v případě, že LS databáze neobsahuje žádný záznam typu 10. Na níže uvedeném obrázku je naznačeno schéma vytváření a stárnutí jednotlivých LSA. Vazby mezi vytvářením, obnovou a vytvářením TLV LSA položek jsou ukázány na obrázku 15.



Obrázek 15.

Další důležitou částí při vytváření Opaque LSA je určení hodnoty Opaque ID. U LSA typu 1 až 8 se získává z IP adresy sítě nebo směrovače. U Opaque ID je ale situace obtížnější. Pole má pouze 24 bitů (prvních 8 bitů pole LSID je rezervováno pro Opaque typ). Hodnotu LSID získám z IP adresy LSA záznamu. Na místo prvního oktetu nastavím hodnotu 230, která určí Opaque typ. Zbylých 24 bitů získám z posledních 3 oktetů IP adresy, která je uložená v LSA záznamu. Z původní IP adresy ve tvaru xxx.xxx.xxx.xxx se stane LSID ve tvaru 230.xxx.xxx.xxx. Využívám LSA typu 10, které se distribuují pouze v rámci 1 oblasti, při správném návrhu adresace v oblasti je změna na prvním oktetu adresy nejméně pravděpodobná, tudíž je nejmenší šance kolize LSID.

```
u_int32_t lsid = ntohl (add.s_addr);
lsid = SET_OPAQUE_LSID (230, GET_OPAQUE_ID(lsid));
tmp.s_addr = htonl (lsid);
```

Obecné schéma vytvoření daného LSA a jeho vložení do databáze vypadá následovně (proměnná TMP reprezentuje pomocnou datovou strukturu adresy, která se uloží do těla LSA):

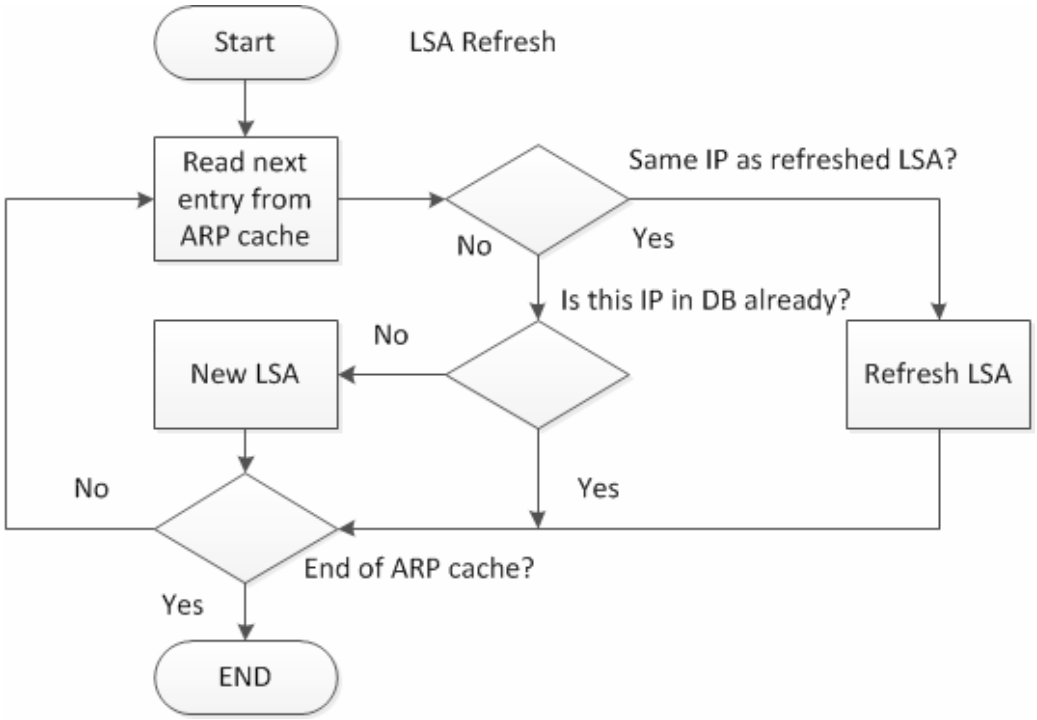
```
new = ospf_tlv_lsa_new_r(area, tmp, MAC);          Vytvoření LSA, nastavení hodnot
new = ospf_lsa_install(area->ospf, NULL, new);     Vložení do databáze
area->ospf->lsa_originate_count++;                 Zvýšení počtu záznamů v DB
ospf_flood_through_area (area, NULL, new);         Zaplavení oblastí daným LSA
```

5.5.3. Refresh jednotlivých LSA

Jelikož záznamy nemohou být libovolně staré, musí se obnovovat. Pro obsluhu obnovení bylo nutné vytvořit metodu, kterou jak jsem již zmínil, musel jsem zaregistrovat do funkční tabulky. Obnova proběhne jednoduše - vytvoří se nová instance LSA záznamu, jejímu tělu se nastaví shodné hodnoty jako má obnovované LSA a tato nová instance se vloží do

databáze. Před vložením nového LSA záznamu do databáze se starý záznam (obnovovaný) musí odstranit ze seznamu záznamů, které se mají obnovovat a odstranit se z tabulek, které se distribuují dále do sítě.

Nicméně pro zajištění periodického prohledání externího zdroje dat (v tomto případě ARP cache operačího systému) jsem musel navrhnout komplexnější mechanismus. Tento mechanismus při každém obnovení libovolného LSA prohledá externí data. Vývojový diagram procesu obnovení je ukázán na obrázku 16.



Obrázek 16.

Na obrázku 16 je znázorněno schéma obnovení každého LSA záznamu. Pokud se daný záznam v ARP cachi již nenachází, nedojde k jeho obnovení a záznam časem z databáze vypadne. Pokud program nalezne záznam, který se v databází ještě nenachází, bude záznam do databáze vložen.

5.5.4. Napojení na ARP cache operačního systému

Spojení s ARP cachí operačního systému jsem realizoval pomocí pomocného textového souboru. Do souboru jsem pomocí systémového volání uložil obsah ARP cache a následně jsem tento soubor prohledával v těle metod Quaggy. Toto jsem v kódu Quaggy realizoval následující instrukcí:

```
system ("arp  -n >  /tmp/arp.txt");
```

Rozhodl jsem se pro tuto možnost implementace z důvodu, že přímý přístup k ARP cachi v jádru operačního systému by mohl způsobit pád systému a taky z důvodu, že přístup k ní by se mohl v různých UNIX-ových OS lišit.

5.5.5. Výpis vlastních LSA na terminál a do souboru

Výpisy jsem se rozhodl spojit s jedním příkazem pro vypsání detailních informací o záznamu LSA typu 10 na virtuální terminál OSPF démona Quaggy. Je to příkaz privilegovaného režimu.

```
#show ip ospf database opaque-area
```

Výpis na terminál probíhá v metodě show_opaque_info_detail (v souboru *ospfd/ospf_opaque.c*). Za detailní výpis informací o hlavičce OSPF a LSA záznamu se vypíše na nový řádek hodnota IP adresy a na další nový řádek se vypíše hodnota MAC adresy.

Výpis do souboru probíhá obdobně, po každém výpisu na terminál se hodnoty IP a MAC adresy vloží na nový řádek textového souboru ve tvaru IP: hodnota MAC: hodnota. Před prvním výpisem jsem musel vyřešit vymazání souboru (soubor před prvním zápisem otevřu s parametrem W, následně pro jednotlivé zápisy soubor otevírám s parametrem A), aby byla vždy v souboru aktuální databáze bez duplicitních záznamů.

6. Testování

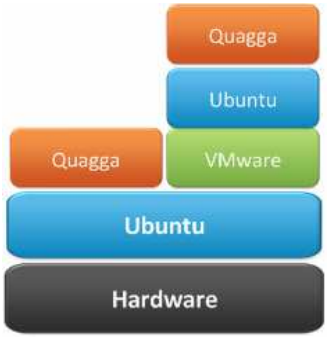
Doposud jsem se zabýval pouze teorií fungování protokolu OSPF, rozborem implementace protokolu a mými úpravami protokolu, nicméně nejdůležitější částí je výsledná funkčnost celého projektu. Tu je nutno pečlivě otestovat a odhalit případné chyby. Testování si vyžádá alespoň dvě samostatné instance Quaggy, takže budu muset využít dva samostatné operační systémy.

Jako první u testování jsem se zaměřil na testování vkládání dat do databáze a na obnovu těchto dat. Při tomto testu jsem si ověřil korektnost registrovaných funkcí a spolehlivost mechanismu vkládání. Pro test obnovení dat jsem musel nastavit hodnotu MaxAge na 20 a obnovovací interval na 10 sekund. Pro testování mechanismu obnovení jsem výstupní soubor příkazu *arp -n* ručně upravoval abych si ověřil zda řídící logika obnovovacího mechanismu korektně zpracovává nově přidané záznamy do vstupního souboru, ze kterého se načítají data, která určují zda se záznam obnoví, smaže nebo vytvoří úplně nové LSA.

Pro testovací účely je topologie se dvěma směrovači dostačující. Řídící logika směrovače má dva hlavní úkoly: informace LSA do sítě odeslat a data ze sítě přijmout, zpracovat a uložit do své databáze.

6.1. Testovací topologie

Jak jsem již zmínil, pro testování jsem potřeboval využít dvou operačních systémů, které by zajistily správnou funkci směrovacích démonů. Rozhodl jsem se zvolit řešení pomocí virtuálního stroje, na kterém jsem provozoval operační systém Ubuntu. Jako hostitelský OS taktéž sloužilo Ubuntu. Tento způsob testování jsem zvolil z důvodu praktičnosti testů. Distribuce kódu je pohodlnější a rychlejší mezi hostitelským a virtualizovaným systémem, než mezi dvěma fyzicky oddělenými systémy. Na obrázku 17 je znázorněná struktura testovacího systému.

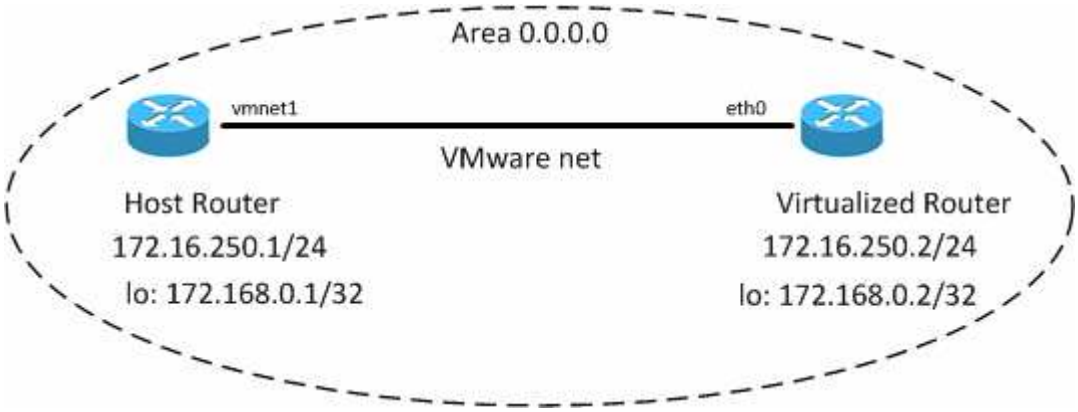


Obrázek 17.

- HW parametry hostitelského systému: Intel Core2Duo T7200 (2 jádra po 2GHz), 4GB DDR2 667MHz.

- HW parametry virtuálního systému: pomocí VMware konfigurace jsem virtuálnímu stroji přiřadil 1 procesorové jádro a 1GB RAM.

Testovací síťovou topologii jsem znázornil na obrázku 18.



Obrázek 18.

Virtualizovaná síť mezi virtuálním a hostitelským strojem je typu Ethertnet. Slouží pro propojení všech virtualizovaných systémů se systémem hostitelským. Tato síť slouží jako vhodný prostředek pro testování a zachytávání provozů mezi dvěma směrovacími aplikacemi Quaggy.

6.2. Konfigurace testovací topologie

Konfiguraci jsem se rozhodl provádět pomocí konfiguračních souborů. Toto řešení je pro testovací účely vhodnější. Konfiguraci si 1 nakonfigurujeme a následně při každém spuštění aplikace provozujeme směrovače s požadovanou konfigurací a nemusíme je konfigurovat pomocí terminálu.

Soubor zebra.conf

```
hostname HostZebra
password z
enable password z

interface lo
 ip address 172.168.0.1/32
 link-detect
 no shutdown

interface vmnet1
```

Soubor zebra.conf

```
hostname VirtualZebra
password z
enable password z

interface lo
 ip address 172.168.0.2/32
 link-detect
 no shutdown

interface eth0
```

```
ip address 172.16.250.1/24
link-detect
no shutdown
```

```
ip address 172.16.250.2/24
link-detect
no shutdown
```

Soubor ospfd.conf

```
hostname HostOspf
password z
enable password z

router ospf
  network 172.168.0.1/32 area 0
  network 172.16.250.0/24 area 0
  capability opaque

log stdout
```

Soubor ospfd.conf

```
hostname VirtualOspf
password z
enable password z

router ospf
  network 172.168.0.2/32 area 0
  network 172.16.250.0/24 area 0
  capability opaque

log stdout
```

Rozhraní typu loopback sloužily pro testování správnosti směrování mezi jednotlivými směrovacími aplikacemi. Testovat konektivitu mezi přímo spojenými rozhraními by neověřilo funkčnost směrovacího procesu. [4]

6.3. Analýza provozu mezi směrovači

Veškerou analýzu provozu jsem prováděl pomocí analyzátoru paketů Wireshark. Zachycené pakety jsem pak exportoval do textového formátu. U detailního výpisu vybraných paketů jsem vypisoval detaily pouze o LSA typu 10.

Jako první při analýze provozu jsem se zaměřil na posloupnost posílaných zpráv.

```
172.16.250.2    224.0.0.5       OSPF    Hello Packet
172.16.250.1    224.0.0.5       OSPF    Hello Packet
172.16.250.2    224.0.0.5       OSPF    Hello Packet
172.16.250.1    172.16.250.2    OSPF    DB Descr.
172.16.250.2    172.16.250.1    OSPF    DB Descr.
172.16.250.1    172.16.250.2    OSPF    DB Descr.
172.16.250.2    172.16.250.1    OSPF    DB Descr.
172.16.250.1    172.16.250.2    OSPF    DB Descr.
172.16.250.1    172.16.250.2    OSPF    LS Request
172.16.250.2    172.16.250.1    OSPF    LS Request
172.16.250.1    224.0.0.5       OSPF    LS Update
172.16.250.2    224.0.0.5       OSPF    LS Update
```

172.16.250.2	224.0.0.5	OSPF	LS Update
172.16.250.2	224.0.0.5	OSPF	LS Update
172.16.250.2	224.0.0.5	OSPF	LS Acknowledge
172.16.250.1	224.0.0.5	OSPF	LS Acknowledge
172.16.250.1	224.0.0.5	OSPF	LS Update
172.16.250.2	224.0.0.5	OSPF	LS Acknowledge

Podle sekvence posílaných zpráv je zřejmé, že komunikace a navázání vztahu přilehlosti probíhá korektně. Jako první se sousední směrovače dostanou až do fáze vztahu Two-way, jak jsem již psal v teortické části, do této fáze se dostanou pomocí mechanismu, který je obdobou 3-fázového handshakingu (schéma na obrázku 5).

Následně se směrovače dohodnou na parametrech komunikace a vymění si popisy svých databází. Pak následuje již vyžádání chybějících LSA. Po přenosu LSA záznamů dojde k jejich potvrzení.

V následujících dílčích podkapitolách se budu věnovat každému typu zpráv, které si směrovače mezi sebou vyměňují. Pro přehlednost uvedu OSPF hlavičku jen u jednoho paketu, jelikož je u všech obdobná a liší se pouze minimálně. U jednotlivých výpisů budu uvádět pouze LSA typu 10, jelikož u ostatních typů předpokládám, že budou fungovat (nijak jsem do jejich mechanismů nezasahoval).

6.3.1. Hello pakety

Hello paket odeslaný z Quaggy, která byla spuštěná na hostitelském systému:

```
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: IPv4mcast_00:00:05
(01:00:5e:00:00:05)
Internet Protocol, Src: 172.16.250.1 (172.16.250.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
   OSPF Header
   OSPF Hello Packet
      Network Mask: 255.255.255.0
      Hello Interval: 10 seconds
      Options: 0x02 (E)
      Router Priority: 1
      Router Dead Interval: 40 seconds
      Designated Router: 0.0.0.0
      Backup Designated Router: 0.0.0.0
```

Nejzajímavější na tomto paketu je to, že i přes povolenou Opaque funkčnost v hello paketu směrovač neoznamuje, že tuto funkci podporuje. Po prostudování dokumentace a technických diskusí jsem zjistil, že je to z důvodu kompatibility s jinými implementacemi

protokolu OSPF. U některých implementací se nenavážou sousedské vazby mezí směrovači, pokud se liší právě v O bitu v poli Options u hello paketů. Komunikace probíhá pomocí multicastové adresy AllSPFRouters.

6.3.2. Database description paket

Paket odeslaný z Quaggy, která byla provozována na virtuálním stroji:

```
Ethernet II, Src: Vmware_58:38:12 (00:0c:29:58:38:12), Dst: Vmware_c0:00:01
(00:50:56:c0:00:01)
Internet Protocol, Src: 172.16.250.2 (172.16.250.2), Dst: 172.16.250.1 (172.16.250.1)
Open Shortest Path First
   OSPF Header
   OSPF DB Description
      Interface MTU: 1500
      Options: 0x42 (O, E)
      DB Description: 0x01 (MS)
      DD Sequence: 1362424778
   LSA Header
   LSA Header
      LS Age: 140 seconds
      Do Not Age: False
      Options: 0x02 (E)
      Link-State Advertisement Type: Opaque LSA, Area-local scope (10)
      Link State ID Opaque Type: Unknown (230)
      Link State ID Opaque ID: 16777215
      Advertising Router: 172.168.0.2 (172.168.0.2)
      LS Sequence Number: 0x80000001
      LS Checksum: 0xb622
      Length: 48
```

Při této fázi komunikace směrovače spolu komunikují přímo, bez využití multicastových adres. Až v tomto typu paketů si směrovače sdělí, že podporují Opaque LSA. Dochází také k dohodnutí Master/slave rolí při výměně databází. Výpis paketu potvrzuje korektní vložení dat do databáze. Směrovač následně informace o něm propaguje svému sousednímu směrovači.

6.3.3. Link state request

Paket odeslaný ze směrovače na hostitelském systému:

```
Frame 13 (82 bytes on wire, 82 bytes captured)
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_58:38:12
(00:0c:29:58:38:12)
Internet Protocol, Src: 172.16.250.1 (172.16.250.1), Dst: 172.16.250.2 (172.16.250.2)
Open Shortest Path First
  OSPF Header
    OSPF Version: 2
    Message Type: LS Request (3)
    Packet Length: 48
    Source OSPF Router: 172.168.0.1 (172.168.0.1)
    Area ID: 0.0.0.0 (Backbone)
    Packet Checksum: 0x6418 [correct]
    Auth Type: Null
    Auth Data (none)
  Link State Request
  Link State Request
    Link-State Advertisement Type: Opaque LSA, Area-local scope (10)
    Link State ID: 230.255.255.255
    Advertising Router: 172.168.0.2 (172.168.0.2)
```

OSPF hlavička neobsahuje žádné překvapivé údaje. Výpis potvrzuje, že délka paketu v bytech musí být bezezbytku dělitelná 4.

Z výpisu je možné vyčíst, že funkčnost podporující Opaque LSA jsem v programu nastavil správně. Směrovač si korektně vyžádá přenos LSA typu 10, hlavička LSA se z DDP zpracuje a je využitá k vyžádání aktualizace databáze. Komunikace probíhá přímo mezi směrovači pomocí unicastových adres.

6.3.4. Link state update

Aktualizace záznamů, které odesílá směrovač provozovaný na virtuálním systému:

```
Ethernet II, Src: Vmware_58:38:12 (00:0c:29:58:38:12), Dst: IPv4mcast_00:00:05
(01:00:5e:00:00:05)
Internet Protocol, Src: 172.16.250.2 (172.16.250.2), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
  OSPF Header
  LS Update Packet
    Number of LSAs: 2
    LS Type: Router-LSA
    LS Type: Opaque LSA, Area-local scope
      LS Age: 141 seconds
```

```
    Do Not Age: False
    Options: 0x02 (E)
    Link-State Advertisement Type: Opaque LSA, Area-local scope (10)
    Link State ID Opaque Type: Unknown (230)
    Link State ID Opaque ID: 16777215
    Advertising Router: 172.168.0.2 (172.168.0.2)
    LS Sequence Number: 0x80000001
    LS Checksum: 0xb622
    Length: 48
    Unknown LSA Type 230
```

Na rozdíl od DDP a LSR paketů, se již odesílá celý LSA záznam, nikoli pouze jeho hlavička. Rozesílání aktualizací databází využívá multicastovou komunikaci (např. v broadcastové síti je to efektivnější než posílat LSA záznam každému směrovači samostatně).

6.3.5. Link state acknowledge

Potvrzení příjmu aktualizace od směrovače z hostitelského systému:

```
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: IPv4mcast_00:00:05 (01:00:5e:00:00:05)
Internet Protocol, Src: 172.16.250.1 (172.16.250.1), Dst: 224.0.0.5 (224.0.0.5)
Open Shortest Path First
  OSPF Header
  LSA Header
  LSA Header
    LS Age: 141 seconds
    Do Not Age: False
    Options: 0x02 (E)
    Link-State Advertisement Type: Opaque LSA, Area-local scope (10)
    Link State ID Opaque Type: Unknown (230)
    Link State ID Opaque ID: 16777215
    Advertising Router: 172.168.0.2 (172.168.0.2)
    LS Sequence Number: 0x80000001
    LS Checksum: 0xb622
    Length: 48
  LSA Header
```

Potvrzení probíhá až po zpracování přijatého LSA, tudíž nám tento paket potvrzuje funkčnost přijetí a zpracování mnou definovaných obecných LSA položek. Závěrečným testem

korektnosti přenesených LSA bude jejich výpis na terminál. Potvrzování přijímaných LSA probíhá taktéž pomocí multicastové komunikace.

6.4. Výpisy a zhodnocení funkčnosti

Jak jsem již zmiňoval dříve, o korektním přenesení načtených dat z ARP cache, se přesvědčíme výpisem přeneseného LSA do konzole. Pro výpis jsem využil již implementovaný příkaz, který vypisuje detailní informace o Opaque LSA záznamech z databáze. Jako ukázku přikládám výpis pouze jednoho LSA záznamu:

```
OSPF Router with ID (172.168.0.2)


        Area-Local Opaque-LSA (Area 0.0.0.0)

  LS age: 7
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x3
  LS Type: Area-Local Opaque-LSA
  Link State ID: 230.168.1.1 (Area-Local Opaque-Type/ID)
  Advertising Router: 172.168.0.1
  LS Seq Number: 80000001
  Checksum: 0x8e41
  Length: 44
  Opaque-Type 230 (TLV-LSA)
  Opaque-ID   0xa80101
  Opaque-Info: 24 octets of data
  IP: 192.168.1.1
  MAC: bc:c8:10:60:35:77
```

Tomuto LSA záznamu odpovídá v exportovaném souboru záznam:

```
IP: 192.168.1.1 , MAC: bc:c8:10:60:35:77
```

Údaje vypsaného LSA odpovídají IP i MAC adrese mého domácího směrovače, prostřednictvím kterého je můj notebook připojen k internetu.

Pomocí posledního testovacího kroku (výpisů) jsem ověřil správnou funkci navrženého a implementovaného řešení. Během všech fází testování (test vkládání, přenosu a výpisu) se Quagga chovala stabilně. Implementací jsem splnil požadavky na ni kladené.

7. Závěr

Tato práce byla pro mně velice zajímavá a přínosná. Její přínos spočíval v několika věcech. Prvním přínosem bylo bezesporu nastudování RFC dokumentu protokolu OSPF. Nabyl jsem spoustu důležitých poznatků, které jsem následně využil při studiu a taky mi tyto vědomosti mohou přinést výhodu v budoucím profesním životě. Jako další oceňuji nabyté zkušenosti a prohloubení znalostí o programovacím jazyce C.

Během analýzy kódu Quaggy jsem si ověřil svůj názor, že orientace v cizím kódu je obtížná, zejména pokud je kód psán několika různými programátory a komentáře jsou velice stručné. Ideálním příkladem špatné orientace v kódu byla tvorba metody pro registraci funkční tabulky mého typu Opaque LSA. Celá inicializace se skládá z volání tří metod, což se může zdát jako jednoduché, ale opak je pravdou. Celkový čas strávený nad prohledáváním kódu a hledáním obecného schématu inicializace libovolného Opaque typu mi zabrala 5 hodin.

Návrh mechanismu na spojení Quaggy s externím zdrojem dat přinesl mé implementaci využitelnost při praktickém nasazení. Pokud bych tuto vlastnost neimplementoval, tak by se práce dala využít jen v laboratorních podmínkách jako ukázka možností Opaque LSA v protokolu OSPF. Jako zdroj dat sloužila ARP cache operačního systému. Z této cache jsem využíval záznamy, které uvádějí jakou IP adresu využívá zařízení s MAC adresou. Tyto statistické informace mohou sloužit jako vstup aplikaci, která by mohla provádět optimalizaci směrování.

Testování jsem prováděl průběžně během vývoje aplikace. Prvními testy jsem se zaměřil na správnost a korektnost přenosu LSA 10 (Opaque type 230) jako celku, nikoli na korektnost přenášených dat. Tyto testy jsem prováděl pomocí analyzátoru síťového provozu. Až následně jsem prováděl testy korektnosti přenášených dat pomocí výpisů přijatých LSA na terminál. Implementace výpisu se řadila do kategorie logiky zpracování přijatých dat na směrovači. Jelikož se data přijatá směrovačem na hostitelském systému shodovala s daty v ARP cachi virtuálního systému, tak testy můžu označit za úspěšné.

Cílem práce bylo vytvořit aplikaci, která bude poskytovat schéma a návod jak postupovat při návrhu aplikace využívající schopnosti implementace protokolu OSPF pro transport obecných dat. Jako testovací datové struktury jsem zvolil dvojici MAC adresa a IP adresa. Tento cíl se mi povedlo splnit.

Literatura

[1] MOY, J. NETWORK WORKING GROUP. OSPF Version 2, RFC2328. 1998. [cit. 2013-04-09] Dostupné z: http://www.ietf.org/rfc/rfc2328.txt

[2] GRYGÁREK, Petr. Směrovací protokol OSPF [online]. [cit. 2013-04-09] Dostupné z: http://www.cs.vsb.cz/grygarek/SPS/lect/OSPF/ospf.html

[3] MOY, John T. OSPF; Anatomy of an Internet Routing Protocol. United States of America: Addison-Wesley, 2000. ISBN 0-201-63472-4.

[4] SCHRODER, Carla. Linux Networking Cookbook. United States of America: O'Reilly Media, Inc., 2007. First Edition. ISBN 0-596-10248-8.

[5] ISHIGURO, Kunihiro et al. Untitled [online]. 2005. [cit. 2013-04-09] Dostupné z: http://www.nongnu.org/quagga/docs/quagga.html

[6] BERGER, L., I. BRYSKIN, A. ZININ a R. COLTUN. NETWORK WORKING GROUP. The OSPF Opaque LSA Option, RFC5250. 2008. [cit. 2013-04-09] Dostupné z: https://tools.ietf.org/html/rfc5250

[7] MOY, J. NETWORK WORKING GROUP. *OSPF Version 2, RFC 1247*. 1991. [cit. 2013-04-09] Dostupné z: http://tools.ietf.org/html/rfc1247

[8] COLTUN, R. *The OSPF Opaque LSA Option, RFC 2370*. 1998. [cit. 2013-04-09] Dostupné z: http://www.ietf.org/rfc/rfc2370.txt

[9] DOSTÁLEK, Libor a Alena KABELOVÁ. *Velký průvodce protokoly TCP/IP a systémem DNS*. Praha: Computer Press, 2000. ISBN 80-7226-323-4.

[10] TANENBAUM, Andrew S. a David J. WETHERHALL. *Computer networks*. United States of America.: Pearson Education, 2011. Fifth edition. ISBN 978-0-13-212695-3.

Obsah CD

Složka	Obsah
/src	Zdrojový kód aplikace.
/text	Text práce ve formátu PDF/A a DOC.